# Essays in Mixed-Integer Nonlinear Optimization for Matching Applications

Pitchaya Wiratchotisatian

A Dissertation

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Doctor of Philosophy

in

Data Science

August 2021

APPROVED:

Professor Andrew C. Trapp (WPI, Advisor)

Professor Hoda Atef Yekta (James Madison University, Co-Advisor)

Professor Jian Zou (WPI)

Professor Gábor N. Sárközy (WPI)

**Abstract**

Mixed-integer optimization (MIO) has seen widespread use in solving challenging decision problems due to its expressivity and ability to characterize constrained decisions under an objective function to be optimized. Advances in algorithmic development and computing over the past several decades have seen profound increases in the potential of commercial optimization solvers [1]. Even so, many real-world problems are nonconvex and nonlinear and representing vexing challenges to be tackled using MIO solvers. The focus of this dissertation is on combining MIO and associated techniques together with machine learning to tackle difficult assignment and matching problems.

First, we study the maximum a-posteriori (MAP) estimation problem for the Gaussian mixture model using mixed-integer nonlinear optimization. In this work, we linearize the nonlinear objective function components using both McCormick relaxations and piecewise linear approximations, which can represent the nonlinearities to within any degree of accuracy. We then seek to accelerate solving this NP-hard optimization problem through a custom Branch and Bound (B&B) algorithm considering multiple branching strategies. Specifically, the branching process of each node represents the fixing of data point(s) based on its integrality. We examine the computational efficiency of our methods, as well as the quality of the generated solutions, and compare with standard machine learning methods for solving the MAP problem.

Second, we propose a decision support system for IT services to ensure that the right tickets get to the right technicians, resulting in improved use of resources. The system embeds machine learning techniques to classify incoming tickets according to service categories, and subsequently uses mixed-integer optimization to assign tickets to technicians based on the service category, available technician capacity, and their skill sets. Computational performance on real-sized data sets reveal that our methods can efficiently assign tickets to technicians.

Third, we study many-to-one matching problems with incomplete preference lists and ties using integer optimization. We study the classical concepts of stability in many-to-one matchings from unique vantage points and so introduce new stability representations, and create algorithms to enhance the computational performance of the forms we introduce. Because there are multiple objective of interest, we lexicographically construct a master objective function that combines each objective component in a strict ordering. Moreover, we study the creation of cohorts to ensure balanced teams according to desirable attributes at project centers. We conduct comprehensive experiments on both simulated data and real data from the WPI student-to-project-center matching process to study the

computational performance of our proposed methods and compare them with the extant literature. Our results reveal favorable performance for matching applications where sufficient seats exist for applicants, such as school choice problems and hospital-residency matching.

The final work in this dissertation explores the solution of a class of integer optimization problems with polynomial objective functions (polynomial integer nonlinear optimization, or PINLO) that is featured in the third work. We introduce a theoretical reformulation that can linearize polynomial functions of separable and bounded integer variables of any degree, and compare it with an alternative reformulation. Computational experiments reveal that our integer linear optimization (ILO) reformulation is computationally tractable for solving large PINLO problems via Gurobi (up to 10,000 constraints and 20,000 variables). This is much larger than current leading commercial global optimization solvers such as BARON, thereby demonstrating its promise for use in real-world applications of integer linear optimization with a polynomial objective function.

**Acknowledgements**

First and foremost, I would like to express my gratitude to my advisor, Professor Andrew C. Trapp, who gave me invaluable advice and continuous support in both academics and life. Thank you very much for his understanding and encouragement over the past four years and his little jokes that brightened my day.

I am grateful to my colleagues, Professor Patrick Flaherty, Ji Ah Lee, Zhou Tang, Christopher J. Chagnon, Professor Soussan Djamasbi, Professor Hoda Atef Yekta, Erin Koontz Bell, Deborah Fusaro, Anne Ogilvie, Mark Anthony Santiago, and Alfred Scott, without their knowledge and assistance, this dissertation thesis would have never been accomplished.

I would like to thank my dissertation committee, including Professor Jian Zou, Professor Gábor N. Sárközy, and Professor Hoda Atef Yekta for their time and advice.

Most importantly, I would like to extend my special thanks to my family and friends for their love and support. This long journey would be too boring without them listening, talking, understanding, and spending their free time with me. I, especially, thank my husband who is my best friend in life and has tried his best to take care of our baby (who always has an urge to help me working on my laptop) while I was working.

I also would like to express my gratitude to everyone, who directly or indirectly, have lent their hand in this chapter of my life. This accomplishment would not have been possible without them. Thank you!

# Contents

# Chapter 1

# Introduction

This dissertation studies mixed-integer nonlinear optimization (MINLO) for matching and assignment applications. The general form of MINLO in our study can be formulated as:

$$\begin{array}{ll} \text{minimize} & f(x,y) \\ \text{such that} & g_i(x,y) \le b_i, \quad i = 1,\dots,q, \\ & h_i(x,y) = 0, \quad i = 1,\dots,\ell, \\ & x \in X \subseteq \mathbb{Z}^m,\ y \in Y \subseteq \mathbb{R}^n, \end{array} \tag{1.1}$$

where $f$, $g_i$, and $h_i$ are real-valued functions, $\mathbb{Z}^m$ denotes the set of integer vectors in $\mathbb{R}^m$, and $X$ is nonempty.

Mixed-integer optimization (MIO) has been widely used in many decision making problems in many areas, such as engineering, management science, operations research, and scientific computing. Some examples of its versatile applications include chemical engineering, knapsack problem, resource allocation, computer networks, and portfolio selection. Yet MIO has been challenging optimization researchers for a long time, due to its difficulties in reasonable computational performance. MIO is known to be NP-hard as it includes mixed-integer linear optimization (MILO) that is known to be NP-hard as a subclass. Its computational complexity increases exponentially as the problem dimension increases. Advances in algorithmic development and computer power during the past several years incredibly increases the potential of optimization solvers [1], opening the opportunity to explore difficult nonconvex and nonlinear MIO problems and apply MIO to solve real-world problems as well as typical statistical and machine learning techniques.

Techniques for global optimization of MIO in general include Branch-and-Bound (B&B), Outer Approximation (OA), and Generalized Benders Decomposition (GBD) algorithms. B&B algorithms recursively split the search space that optimizes the objective and keeps track of bounds of the objective to eliminate inferior regions of the search space by relaxing integrity of integer variables. On the other hand, OA and GBD algorithms iteratively

solve a mixed-integer linear programming master problem and nonlinear programming subproblems. We solve MINLO problems in this dissertation through the B&B algorithm.

This dissertation focuses on using MIO together with machine learning techniques to tackle multiple matching applications, i.e. matching data points into clusters, matching tasks to workers, and matching students to schools/projects. Our work features both theoretical study on MIO and applications of MIO for solving real-world matching problems: (1) We solve the maximum a-posteriori (MAP) estimation problem for the Gaussian mixture model (GMM) using MINLO. We reformulate MINLO to mixed-integer quadratic optimization (MIQO) and customized B&B algorithm considering a specific branching strategy. (2) We propose a decision support system for IT services to ensures that the right tickets get to the right technicians using MIO and machine learning techniques. (3) We introduce new stability representations for solving many-to-one matching problems with incomplete lists and ties using integer optimization (IO), then compare their performance to existing forms in the literature. Additionally, our lexicographic multi-objective functions optimize each objective component in a strict ordering. (4) We linearize the polynomial integer quadratic programming (PIQP) of square deviation penalty objective function in our third work. The reformulation can be extended to a class of polynomial functions of bounded integer variables with any degrees.

The organization of this dissertation is as follows. Chapter 2 presents MAP Clustering via MINLO. Chapter 3 presents the decision support system for IT services that classifies and assigns tickets through MIO and machine learning techniques. Chapter 4 studies the use of IO in a comparative study of stability representations for solving many-to-one matching problems with ties and incomplete preference lists. Finally, Chapter 5 covers the reformulation to linearize polynomial integer nonlinear optimization (PINLO) on a class of polynomial functions of separable and bounded integer variables.

# Chapter 2

# MAP Clustering via Mixed-Integer Nonlinear Optimization

## Introduction

Mixed-integer nonlinear optimization (MINLO) has been used to model a wide variety of applications, such as in chemical engineering, finance, and manufacturing. It is capable to find the global optima with certificate of optimality, though the complexity is NP-hard in general. In this Chapter, we study solving Maximum a-posteriori Estimation (MAP) for Gaussian mixture model (GMM) using MINLO.

GMM represents a mixture of finite normal distributions with unknown parameters $\boldsymbol{\theta}$. It is often used for clustering data by finding a mixture of Gaussian probability distributions that best describe the input dataset. Many techniques exist to learn parameters of mixture models focus on maximum likelihood such as Expectation-Maximization (EM), Maximum a-postiori Estimation (MAP), and Markov Chain Monte Carlo (MCMC). This study focuses on using MAP for clustering observed data into a fixed number of mixture components.

MAP is a method used for estimate unknown parameters. Unlike maximum likelihood estimation (MLE) that estimates parameters using likelihood, MAP incorporates a prior knowledge to estimate unknown parameters, which are obtained from the mode of the posterior distribution. (That is MLE is a special case of MAP where the prior is uniform.) There are several ways to solve MAP such as Expectation-Maximization Algorithm (EM), Variational EM, Sequential Least-Squares Quadratic Programming, and Simulated Annealing. These techniques, however, only guarantee that the parameter estimates will converge to a local optimum. The outcome of some coordinate search algorithms, such as EM, depends on the starting point. For applications where encoding prior information benefits the outcome, such as clinical trials and DNA sequencing, it is

often of interest to find a globally optimal solution. Therefore, this study formulates and solves MAP for GMM using MINLO which, in contrast to other methods, can find global optima and provide strict upper and lower bounds on the global optima through the use of Branch-and-Bound algorithm (B&B).

Significant progress has been made in research with several MINLO solvers available, yet the development in creating reasonable representations as well as computational runtime challenges researchers until now. We reformulate our MINLO to a mixed-integer quadratic optimization (MIQO). This is carried out by using piecewise linear functions to approximate the nonlinear objective function component and using McCormick envelopes to linearize bilinear components of MAP. Our MIQO still requires significant runtime to prove global optimality. We further attempt to improve computational efficiency through custom Branch and Bound process (B&B).

## 2.1 Formulating MAP via MINLO Optimization

Our GMM finds unknown parameters $\boldsymbol{\theta}$ of $K$ normal distributed subgroups that best explain $N$ observations in a dataset. Define $\boldsymbol{y} = (\boldsymbol{y}_1^T, \ldots, \boldsymbol{y}_N^T)^T$ to be observations, each is in $D$-dimensional. Define $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_K)^T$ to be component proportion such that $\sum_{k=1}^{K} \pi_k = 1$ and $\pi_k \geq 0$.

Define the mixture density functions of each observation as $Z_i \overset{i.i.d.}{\sim} \text{Categorical}(\boldsymbol{\pi})$, for $i \in \{1, \ldots, N\}$. Each observation $i$ belongs to exactly one subgroup, that is, $z_i \in \{1, \ldots, K\}$, $z_i = \{0, 1\}^K$. The associated distribution over observations in each subgroup is defined as $Y_i | z_i, \boldsymbol{\mu}_{z_i} \sim \mathcal{N}(\boldsymbol{\mu}_{z_i}, \boldsymbol{\Sigma})$, where the component mean $\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K)$ and the component covariance $\boldsymbol{\Sigma} = (\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K)$. The covariance in our primary model are fixed, hence the unknown parameters in our initial model are component assignment of each observation, component mean, and component proportion, that is, $\boldsymbol{\theta} = \{\boldsymbol{z}, \boldsymbol{\mu}, \boldsymbol{\pi}\}$.

The unknown parameters can be estimated using the maximum log-likelihood of posterior density function. Following Bayes' rule, our objective function is defined as

$$\begin{aligned} \max_{\boldsymbol{z},\boldsymbol{\pi},\boldsymbol{\mu}} \ \log \Pr(\boldsymbol{z}, \boldsymbol{\pi}, \boldsymbol{\mu} | \boldsymbol{y}) \ &\propto \ \max_{\boldsymbol{z},\boldsymbol{\pi},\boldsymbol{\mu}} \ \log \Pr\left(\boldsymbol{y} \mid \boldsymbol{\mu}, \boldsymbol{z}\right) \Pr\left(\boldsymbol{z} \mid \boldsymbol{\pi}\right) \Pr\left(\boldsymbol{\mu}, \boldsymbol{\pi}\right) \\ &\propto \ \max_{\boldsymbol{z},\boldsymbol{\pi},\boldsymbol{\mu}} \ \log \Pr\left(\boldsymbol{y} \mid \boldsymbol{\mu}, \boldsymbol{z}\right) + \log \Pr\left(\boldsymbol{z} \mid \boldsymbol{\pi}\right) + \log \Pr\left(\boldsymbol{\mu}, \boldsymbol{\pi}\right). \end{aligned}$$

We next demonstrate each component in the log objective function. Note that we specify the uniform prior $\Pr(\boldsymbol{\mu}, \boldsymbol{\pi}) \propto 1$, hence we dispose of the third component from the consideration. We assume equivalent variances across all components, which can be estimated by averaging the standard deviations from the given data. The first component

represents least-square loss, which can be calculated as:

$$\begin{aligned}\log \Pr\left(\boldsymbol{y} \mid \boldsymbol{\mu}, \boldsymbol{z}\right) &= \log \textstyle\prod_{i=1}^{N} \Pr\left(\boldsymbol{y}_i \mid \boldsymbol{\mu}, \boldsymbol{z}_i\right) \\ &= \log \textstyle\prod_{i=1}^{N} \prod_{k=1}^{K} \left[(2\pi)^{\frac{D}{2}} |\boldsymbol{\Sigma}_k|^{-\frac{1}{2}} \exp\left(-\tfrac{1}{2}(\boldsymbol{y}_i - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{y}_i - \boldsymbol{\mu}_k)\right)\right]^{z_{ik}} \\ &\propto -\textstyle\sum_{i=1}^{N} \sum_{k=1}^{K} \frac{1}{2} z_{ik} (\boldsymbol{y}_i - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{y}_i - \boldsymbol{\mu}_k), \ (\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_1 = \cdots = \boldsymbol{\Sigma}_K).\end{aligned}$$

The second term represents cross entropy loss, which can be calculated as:

$$\begin{aligned}\log \Pr\left(\boldsymbol{z} \mid \boldsymbol{\pi}\right) &= \log \textstyle\prod_{i=1}^{N} \Pr\left(\boldsymbol{y}_i \mid \boldsymbol{\mu}, \boldsymbol{z}_i\right) \\ &= \log \textstyle\prod_{i=1}^{N} \prod_{k=1}^{K} \pi_k^{z_{ik}} \\ &= \textstyle\sum_{k=1}^{K} \sum_{i=1}^{N} z_{ik} \log \pi_k,\end{aligned}$$

where $\sum_{i=1}^{N} z_{ik}$ represents the number of observations in each component. As the result, our objective function minimizes *least-square loss* and *cross entropy loss*:

$$\min_{\boldsymbol{z}, \boldsymbol{\pi}, \boldsymbol{\mu}} \ \eta \sum_{i=1}^{N} \sum_{k=1}^{K} z_{ik} (\boldsymbol{y}_i - \boldsymbol{\mu}_k)^T (\boldsymbol{y}_i - \boldsymbol{\mu}_k) - \sum_{k=1}^{K} \left( \sum_{i=1}^{N} z_{ik} \right) \log \pi_k$$

where $\eta = \frac{1}{2}\boldsymbol{\Sigma}^{-1}$ representing precision.

Thus, we can formulate MAP as MINLO as follows:

$$\min_{\boldsymbol{z}, \boldsymbol{\pi}, \boldsymbol{\mu}} \quad \eta \sum_{i=1}^{N} \sum_{k=1}^{K} z_{ik} (\boldsymbol{y}_i - \boldsymbol{\mu}_k)^T (\boldsymbol{y}_i - \boldsymbol{\mu}_k) - \sum_{k=1}^{K} \left( \sum_{i=1}^{N} z_{ik} \right) \log \pi_k \tag{2.1a}$$

$$\text{such that} \quad \sum_{k=1}^{K} \pi_k = 1, \tag{2.1b}$$

$$\sum_{k=1}^{K} z_{ik} = 1, \quad \forall\ i = 1, \ldots, N, \tag{2.1c}$$

$$\boldsymbol{\mu}_k^L \leq \boldsymbol{\mu}_k \leq \boldsymbol{\mu}_k^U, \quad \forall\ k = 1, \ldots, K, \tag{2.1d}$$

$$\frac{1}{N} \leq \pi_k \leq \frac{N - K + 1}{N}, \quad \forall\ k = 1, \ldots, K \tag{2.1e}$$

$$z_{ik} \in \{0, 1\}, \quad \forall\ i = 1, \ldots, N,\ \forall\ k = 1, \ldots, K. \tag{2.1f}$$

Objective function (2.1a) minimizes total least-square loss and entropy loss. Probability sums to one in (2.1b). Each observation is assigned to exactly one component in (2.1c). Constraints (2.1d) and (2.1e) specify bounds on component means and component proportions, respectively, where $\boldsymbol{\mu}_k^L$ and $\boldsymbol{\mu}_k^U$ are lower and upper bounds of $\boldsymbol{\mu}_k$ in each of the $D$-dimension, and all components must contain at least one observation. Lastly, variable domain is defined in (2.1f).

Our MINLO may require insufficient computational runtime even on the current lead-

ing MINLO solver. We present two approaches to improve efficiency of solving MINLO in the next sections.

## 2.2 Improving Efficiency of MINLO

MINLO is widely used in many applications, such as chemical engineering, process engineering, power/electrical grids, etc. We solve MINLO for MAP using Branch and Bound algorithm (B&B) which guarantees global optimal outcome with a proximity of the upper and lower bounds. Although significant progress has been made in research, it is still challenging to solve MINLO due to difficulty in modeling, as well as computational runtime. First, we reformulate formulation (2.1) to MIQO using piecewise linear approximation and McCormick relaxations, then we custom Branch-and-Bound (B&B) algorithm.

### 2.2.1 Reformulating MINLO to MIQO via Piecewise Linear Approximation Models

We first approximate the nonlinear objective function component $\log \pi_k$ using piecewise linear approximation (PWL). The value of a nonlinear function $v_k = \log \pi_k$ can be approximated by incorporating the possible evaluated solution along the domain $\pi_k$ with a set of constraints. We can increase the degree of accuracy of $\log \pi_k$ by varying the number of breakpoints. Commonly used PWL model approaches presented in [2] include convex combination model (CC), multiple choice model (MC), and logarithmic model (Log). We next present the formulation of each model.

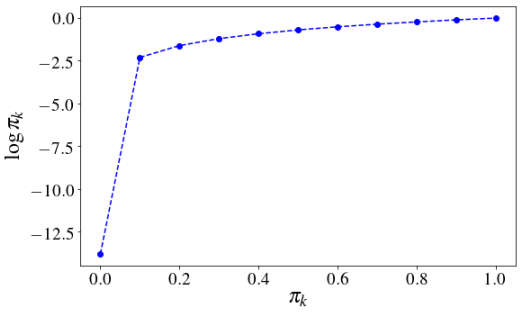



Figure 2.1: Plot of $v_k = \log \pi_k$.

#### Convex Combination Model

CC model approximates the nonlinear function by formulating piecewise linear segments in terms of convex combination of two consecutive breakpoints $b_k^{j-1}$ and $b_k^j$. That is, $\pi_k$ is expressed as $\lambda_k^{j-1} b_k^{j-1} + \lambda_k^j b_k^j$ if $\pi_k \in \left[b_k^{j-1}, b_k^j\right]$. We present the CC model in formulation (2.2) and the notation used in the model in Table 2.1.

| Sets | |
|---|---|
| $\mathcal{I}$ | Set of samples, indexed by $i = 1, \ldots, N$ |
| $\mathcal{K}$ | Set of components, indexed by $k = 1, \ldots, K$ |
| $\mathcal{J}$ | Set of breakpoints $b_k^j$ of size $p$ along x-axis of $v_k$, indexed by $j = 0, \ldots, p$ where $\frac{1}{N} = b_k^0 \leq b_k^1 \leq \cdots \leq b_k^p = \frac{N-K+1}{N}$ $\forall\ k = 1, \ldots, K$ |
| **Decision Variables** | |
| $\lambda_k^j$ | Convex coefficients representing $\pi_k$ $\forall\ k = 1, \ldots, K$, $\forall\ j = 0, \ldots, p$ $\pi_k = \lambda_k^{j-1} b_k^{j-1} + \lambda_k^j b_k^j$ if $\pi_k \in \left[b_k^{j-1}, b_k^j\right]$ |
| $g_k^j$ | Binary variables indicating the location of $v_k$ $\forall\ k = 1, \ldots, K,\ \forall\ j = 1, \ldots, p$ $g_k^j = 1$ if $\pi_k \in \left[b_k^{j-1}, b_k^j\right]$; 0 otherwise |

Table 2.1: Notation for convex combination model.

$$\sum_{j=0}^{p} \lambda_k^j b_k^j = \pi_k, \quad \forall\ k = 1, \ldots, K, \tag{2.2a}$$

$$\sum_{j=0}^{p} \lambda_k^j \log(b_k^j) = v_k, \quad \forall\ k = 1, \ldots, K, \tag{2.2b}$$

$$\sum_{\ell=j}^{p} \lambda_k^\ell \leq \sum_{\ell=j}^{p} g_k^\ell, \quad \forall\ k = 1, \ldots, K,\ \forall\ j = 1, \ldots, p, \tag{2.2c}$$

$$\sum_{\ell=0}^{j-1} \lambda_k^\ell \leq \sum_{\ell=1}^{j} g_k^\ell, \quad \forall\ k = 1, \ldots, K,\ \forall\ j = 1, \ldots, p, \tag{2.2d}$$

$$\sum_{j=0}^{p} \lambda_k^j = 1, \quad \forall\ k = 1, \ldots, K, \tag{2.2e}$$

$$\sum_{j=1}^{p} g_k^j = 1, \quad \forall\ k = 1, \ldots, K, \tag{2.2f}$$

$$\lambda_k^j \geq 0,\ g_k^j \in \{0, 1\}, \quad \forall\ k = 1, \ldots, K,\ \forall\ j = 0, \ldots, p, \tag{2.2g}$$

$$g_k^j \in \{0, 1\}, \quad \forall\ k = 1, \ldots, K,\ \forall\ j = 1, \ldots, p. \tag{2.2h}$$

Constraint sets (2.2a) and (2.2b), respectively, express $\pi_k$ and $v_k$ as convex combination of breakpoints. Constraint sets (2.2c) and (2.2d) together ensure that $pi_k$ is a convex combination of two consecutive breakpoints $b_k^{j-1}$ and $b_k^j$. Constraint set (2.2e) ensures that coefficients of convex combination sum up to one. Constraint set (2.2f) ensures that $v_k$ lies in exactly one piecewise linear segment. Lastly, nonnegativity and binary

restrictions are defined in (2.2g) and (2.2h), respectively.

### Multiple Choice Model

MC model approximates the nonlinear function by formulating piecewise linear segments in terms of linear functions. When $\pi_k \in \left[b_k^{j-1}, b_k^j\right]$, $v_k$ can be approximated as $\Delta_k^j \omega_k + c_k^j$, where $\Delta_k^j$ and $c_k^j$ are the value of slope and intersect of the $j^{\text{th}}$ segment, respectively. We present the MC model in formulation (2.3) and additional notation used in the model in Table 2.2.

| | |
|---|---|
| Parameters | |
| $\Delta_k^j$ | Slopes of the $j^{\text{th}}$ segment $\forall\ k = 1, \ldots, K$, $\forall\ j = 1, \ldots, p$; $\Delta_k^j = \frac{\log\left(b_k^j\right) - \log\left(b_k^{j-1}\right)}{b_k^j - b_k^{j-1}}$ |
| $c_k^j$ | Intersects of the $j^{\text{th}}$ segment $\forall\ k = 1, \ldots, K$, $\forall\ j = 1, \ldots, p$ |
| Decision Variables | |
| $\omega_k^j$ | Values of $\pi_k$ when $\pi_k$ is in the $j_{\text{th}}$ segment $\forall\ k = 1, \ldots, K$, $\forall\ j = 1, \ldots, p$<br>$\omega_k^j = \pi_k$ if $\pi_k \in \left[b_k^{j-1}, b_k^j\right]$; 0 otherwise |

Table 2.2: Additional notation for multiple choice model.

$$\sum_{j=1}^{p} \omega_k^j = \pi_k, \quad \forall\ k = 1, \ldots, K, \tag{2.3a}$$

$$\sum_{j=1}^{p} \left(\Delta_k^j \omega_k^j + c_k^j g_k^j\right) = v_k, \quad \forall\ k = 1, \ldots, K, \tag{2.3b}$$

$$\sum_{j=1}^{p} g_k^j = 1, \quad \forall\ k = 1, \ldots, K, \tag{2.3c}$$

$$b_k^{j-1} g_k^j \leq \omega_k^j \leq b_k^j g_k^j, \quad \forall\ k = 1, \ldots, K,\ \forall\ j = 1, \ldots, p, \tag{2.3d}$$

$$g_k^j \in \{0, 1\}, \quad \forall\ k = 1, \ldots, K,\ \forall\ j = 1, \ldots, p. \tag{2.3e}$$

Constraint set (2.3a) enforces the value of $\omega_k^j$ to be equal to $\pi_k^j$ when $\pi_k$ lies in $\left[b_{k-1}^j, b_k^j\right]$. Constraint set (2.3b) expresses $v_k$ as a linear function of $\Delta_k^j$ and $c_k^j$ when $\pi_k$ lies in $\left[b_{k-1}^j, b_k^j\right]$. Constraint sets (2.3d) and (2.3e) define the value of $\omega_k^j$ and $g_k^j$, respectively.

### Logarithmic Model

Similar to CC model, Log model approximates the nonlinear function by formulating piecewise linear segments in terms of convex combination, however, it applies $n$-bit Gray codes, also known as reflective binary codes, for an independent branching scheme to reduce the size of variables and constraints in CC model. The number of binary variables $g_k^j$ becomes $\lceil \log_2 p \rceil$, which is the length of binary Gray codes needed to construct disjunctive constraints, whereas the number of constraints for $\lambda_k^j$ becomes $2\lceil \log_2 p \rceil$. For instance, if the value of the nonlinear function $v_k$ is approximated using four segments, CC model requires four binary variables $g_k^j$ and eight constraints for $\lambda_k^j$ to represent each segment in which only one of them includes the value of $v_k$, whereas Log model requires only two binary variables $g_k^j$ and four disjunctive constraints for $\lambda_k^j$. Figure 2.2 illustrates 4-bit reflective binary code which can be used to construct up to 16 segments.

| bits | 4 | 3 | 2 | 1 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 1 |
| 3 | 0 | 0 | 1 | 0 |
| 4 | 0 | 1 | 1 | 0 |
| 5 | 0 | 1 | 1 | 1 |
| 6 | 0 | 1 | 0 | 1 |
| 7 | 0 | 1 | 0 | 0 |
| 8 | 1 | 1 | 0 | 0 |
| 9 | 1 | 1 | 0 | 1 |
| 10 | 1 | 1 | 1 | 1 |
| 11 | 1 | 1 | 1 | 0 |
| 12 | 1 | 0 | 1 | 0 |
| 13 | 1 | 0 | 1 | 1 |
| 14 | 1 | 0 | 0 | 1 |
| 15 | 1 | 0 | 0 | 0 |

Figure 2.2: Reflective Binary Code

We present the Log model in formulation (2.4) and additional notation used in the model in Table 2.3. We demonstrate the use of Gray codes for constructing disjunctive constraints in Example 1.

| Sets | |
|---|---|
| $\mathcal{S}$ | Set of indices of left and right breakpoints of each segment, indexed by $j = 1, \ldots, p-1$ $S_j := \{j-1, j\}$ |
| $L_s$ | Set of indices of breakpoints in $S_j$ considering the $s^{\text{th}}$ entry of Gray codes of value 0 $L_s := \cup_{j \in \mathcal{J}, j \notin \sigma(B_s(j))} S_j$; used for indices of $\lambda_k^j$ |
| $R_s$ | Set of indices of breakpoints in $S_j$ considering the $s^{\text{th}}$ entry of Gray codes of value 1 $R_s := \cup_{j \in \mathcal{J}, j \in \sigma(B_s(j))} S_j$; used for indices of $\lambda_k^j$ |
| **Parameters** | |
| $B(j)$ | The $j^{\text{th}}$ reflective binary Gray code representing the $j^{\text{th}}$ segment |
| $B_s$ | The $s^{\text{th}}$ entry of the $j^{\text{th}}$ Gray code $B(j)$ |
| $\sigma(B_s)$ | Support of $B_s$ where $\sigma(B_s) = \{j \in \mathcal{J} : B_s(j) \neq 0\}$ |

Table 2.3: Additional notation for logarithmic model.

$$\sum_{j=0}^{p} \lambda_k^j b_k^j = \pi_k, \quad \forall\ k = 1, \ldots, K, \tag{2.4a}$$

$$\sum_{j=0}^{p} \lambda_k^j \log(b_k^j) = v_k, \quad \forall\ k = 1, \ldots, K, \tag{2.4b}$$

$$\sum_{j \notin L_s} \lambda_k^j \leq g_k^s, \quad \forall\ k = 1, \ldots, K,\ \forall\ s = 1, \ldots, \lceil \log_2 p \rceil, \tag{2.4c}$$

$$\sum_{j \notin R_s} \lambda_k^j \leq (1 - g_k^s), \quad \forall\ k = 1, \ldots, K,\ \forall\ s = 1, \ldots, \lceil \log_2 p \rceil, \tag{2.4d}$$

$$\sum_{j=0}^{p} \lambda_k^j = 1, \quad \forall\ k = 1, \ldots, K, \tag{2.4e}$$

$$\sum_{j=1}^{p} g_k^j = 1, \quad \forall\ k = 1, \ldots, K, \tag{2.4f}$$

$$\lambda_k^j \geq 0, \quad \forall\ k = 1, \ldots, K,\ \forall\ j = 0, \ldots, p, \tag{2.4g}$$

$$g_k^j \in \{0, 1\}, \quad \forall\ k = 1, \ldots, K,\ \forall\ s = 1, \ldots, \lceil \log_2 p \rceil \tag{2.4h}$$

All constraint sets are defined in a similar manner as in CC model formulation (2.2), except that constraint sets (2.4c) and (2.4d) together construct disjunctive constraints of each segment.

**Example 1** *Constructing disjunctive constraints of four segments using 2-bit Gray codes.*

Figure 2.3 shows gray codes corresponding to each of the four segments $B(j)$ where the set of indices of each segment $S_j$ is defined to be $\{j-1, j\}$. Figure 2.4 demonstrates the construction of disjunctive constraints for the convex combination each segment.

| $j$ | $S_j$ | $B(j)$ | |
|---|---|---|---|
| 1 | {0,1} | 0 | 0 |
| 2 | {1,2} | 0 | 1 |
| 3 | {2,3} | 1 | 1 |
| 4 | {3,4} | 1 | 0 |

Figure 2.3: Gray codes of 4-segment.

Each constraint is constructed by considering each entry of the Gray codes. For instance, $L_1$ is the union of breakpoints of $S_j$'s such that the first entry of Gray codes $B(j)$ is zero; as the result, $L_1 = S_1 \cup S_2 = \{0, 1, 2\}$. On the other hand, $R_1$ is the union of breakpoints of $S_j$'s such that the first entry of $B(j)$ is one; as the result, $R_1 = S_3 \cup S_4 = \{2, 3, 4\}$.

By considering the first entry of the Gray codes, we obtain two constraints: $\lambda_3^j + \lambda_4^j \leq g_k^1$ and $\lambda_0^j + \lambda_1^j \leq 1 - g_k^1$. On the other hand, by considering the second entry of the Gray codes, we obtain another two constraints: $\lambda_2^j \leq g_k^2$ and $\lambda_0^j + \lambda_4^j \leq 1 - g_k^2$. The set of left and right breakpoints of the first segment is $S_1 = \{0, 1\}$. By substituting the first and second entries of Gray code corresponding to the first segment $(0, 0)$ to $g_k^1$ and $g_k^2$, respectively, we obtain the following constraints: $\lambda_3 + \lambda_4 \leq 0$, $\lambda_0 + \lambda_1 \leq 1$, $\lambda_2 \leq 0$, and $\lambda_0 + \lambda_4 \leq 1$; all together imply that $\lambda_0 + \lambda_1 = 1$, that is, the value of $v_k$ lies in between the first segment.

| $s$ | $L_s$ | $\sum_{j \notin L_s} \lambda_k^j \leq g_k^s$ | $R_s$ | $\sum_{j \notin R_s} \lambda_k^j \leq 1 - g_k^s$ |
|---|---|---|---|---|
| **1** | {0,1,2} | $\lambda_3 + \lambda_4 \leq g_k^1$ | {2,3,4} | $\lambda_0 + \lambda_1 \leq 1 - g_k^1$ |
| **2** | {0,1,3,4} | $\lambda_2 \leq g_k^2$ | {1,2,3} | $\lambda_0 + \lambda_4 \leq 1 - g_k^2$ |

(a) Constructing disjunctive constraints from each entry of the Gray codes.

| $g_k^1$ | $g_k^2$ | Selected $S_j$ |
|---|---|---|
| 0 | 0 | {0,1} |
| 0 | 1 | {1,2} |
| 1 | 0 | {2,3} |
| 1 | 1 | {3,4} |

(b) Different values of $\left(g_k^1, g_k^2\right)$ result in different segments.

Figure 2.4: Constructing disjunctive constraints from Gray codes.

MC model often performs best on small number of breakpoints; for larger number of breakpoints, Log model is preferable [2]. As different PWL models result in comparatively similar values of $v_k$, we implement PWL using MC model. We next apply McCormick relaxations to linearize bilinear components in the objective function.

### 2.2.2 Reformulating MINLO to MIQO via McCormick Relaxations

McCormick [3] proposed the relaxation of $z = x_1 x_2$, where $x_1 \in [l_1, u_1]$ and $x_2 \in [l_2, u_2]$, with the following inequalities:

$$z \geq u_2 x_1 + u_1 x_2 - u_1 u_2 \quad z \geq l_2 x_1 + l_1 x_2 - l_1 l_2 \tag{2.5a}$$

$$z \leq u_2 x_1 + l_1 x_2 - l_1 u_2 \quad z \leq l_2 x_1 + u_1 x_2 - u_1 l_2 \tag{2.5b}$$

Our final step to convert MINLO to MIQO is to apply McCormick relaxations [3] to linearize the nonlinear components $z_{ik}\mu_{kd}$ and $z_{ik}v_k$ where $v_k = \log \pi_k$. Define $y_{id}$ and $\mu_{kd}$ to be the $d^{\text{th}}$ dimensional element of $\boldsymbol{y}_i$ and $\boldsymbol{\mu}_k$, respectively. We present McCormick inequalities for linearing $z_{ik}\mu_{kd}$ as $t_{ikd}$ and $z_{ik}v_k$ as $w_{ik}$, $\forall\, i = 1, \ldots, N$, $\forall\, k = 1, \ldots, K$, $\forall\, d = 1, \ldots, D$, in constraint sets (2.6) and (2.7), respectively.

$$\begin{aligned} \min_i\{y_{id}\} z_{ik} \leq t_{ikd} \leq \max_i\{y_{id}\} z_{ik} \\ \mu_{kd} - \max_i\{y_{id}\}\,(1 - z_{ik}) \leq t_{ikd} \leq \mu_{kd} - \min_i\{y_{id}\}\,(1 - z_{ik}) \end{aligned} \tag{2.6}$$

$$\begin{aligned} \log \frac{1}{N} z_{ik} \leq w_{ik} \leq \log \frac{N-K+1}{N} z_{ik} \\ v_k - \log \frac{N-K+1}{N}\,(1 - z_{ik}) \leq w_{ik} \leq v_k - \log \frac{1}{N}\,(1 - z_{ik}) \end{aligned} \tag{2.7}$$

In consequence, we obtain the following objective function for our MIQO model:

$$\min_{\boldsymbol{z},\boldsymbol{\pi},\boldsymbol{\mu}} \quad \eta \sum_{i=1}^{N} \sum_{d=1}^{D} \left( y_{id} - \sum_{k=1}^{K} t_{ikd} \right)^2 - \sum_{i=1}^{N} \sum_{k=1}^{K} w_{ik}. \tag{2.8}$$

We next custom B&B algorithm to improve solving efficiency of MIQO.

### 2.2.3 Customizing Branch-and-Bound Algorithm

We solve our MIQO using Branch-and-Bound (B&B), which is a divide-and-conquer algorithm. To find an optimal solution that minimizes the objective function, B&B recursively splits the search space and keeps track of bounds of the objective to eliminate inferior regions of the search space. It exploits the efficient solution of convex relaxations to prune the tree of unpromising regions.

B&B recursively relaxes the component assignments $z_{ik}$ at each node, $z_{ik} \in [0, 1]$. It then branches on integer values of $z_{ik} \in \{0, 1\}$ which are chosen based on some branching strategy. We choose to branch on most integral $z_{ik}$ which forces one branch to be $z_{ik} = 1$ and the other branch to be $z_{ik} = 0$. We expect the branch with $z_{ik} = 0$ willbe fathomed more quickly. The algorithm keeps track of the best solution found until it terminates. We next present the experiment of our MIQO on three UCI datasets.

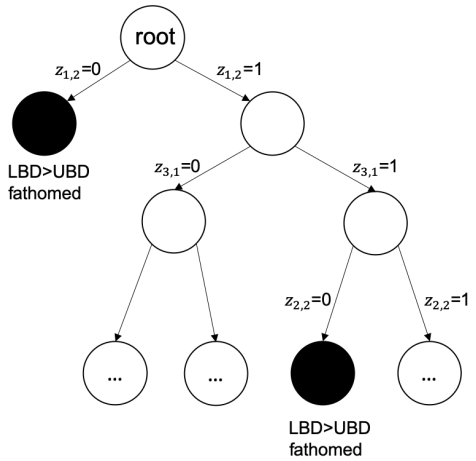



Figure 2.5: An illustration of the (ideal) branching strategy on $z_{ik}$ closest to one.

## 2.3 Computational Experiments

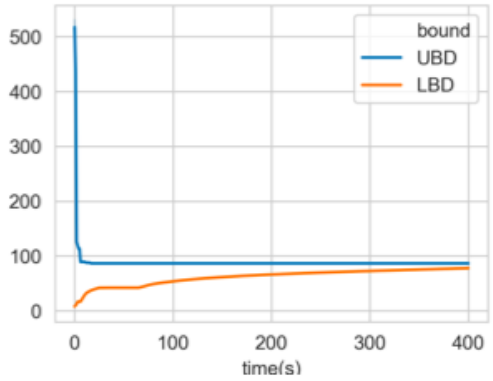
We compare our MIQO with MINLO and EM. We experiment on three famous datasets from UCI Machine Learning Repository [4]: Iris projecting on the first principal component where $N = 150$ and $K = 3$; Wine Quality where $N = 178$, $K = 3$, and $D = 13$;

Wisconsin Breast Cancer where $N = 569$, $K = 2$, and $D = 3$. The PWL approximation of $\log \pi_k$ uses MC model with 5 breakpoints. We explored the most-integral branching strategy, using our customized B&B adapt from [5], but the performance was inferior to the state-of-the-art solvers. As a result, we use GAMS for implementation and the results consist of these two approaches: we use B&B to solve MINLO in formulation (2.1) by calling BARON at each node and use B&B to solve MIQO by calling Gurobi at each node. We limit the computational time to 12 hours.

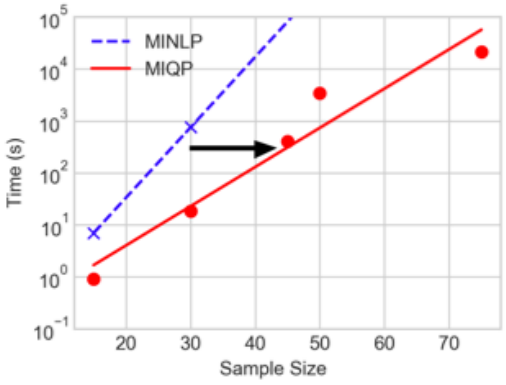
| Data | Metric | EM | MINLO | MIQO |
|---|---|---|---|---|
| iris-1d | Objective Value | 280.6 | **280.02** | 282.71 |
| | Best Bound | - | 9.27 | **161.6** |
| | $\sup\lvert\boldsymbol{\pi} - \hat{\boldsymbol{\pi}}\rvert$ | 0.075 | 0.093 | 0.165 |
| | $\lVert\boldsymbol{\mu} - \hat{\boldsymbol{\mu}}\rVert_2$ | 0.278 | 0.356 | 0.356 |
| | $\sum_i \sup\lvert\boldsymbol{z}_i - \hat{\boldsymbol{z}}_i\rvert/N$ | 0.067 | 0.093 | 0.093 |
| wine-13d | Objective Value | 1376.00 | **1366.85** | 1390.13 |
| | Best Bound | - | $-2.2 \times 10^5$ | **183.42** |
| | $\sup\lvert\boldsymbol{\pi} - \hat{\boldsymbol{\pi}}\rvert$ | 0.005 | 0.006 | 0.167 |
| | $\lVert\boldsymbol{\mu} - \hat{\boldsymbol{\mu}}\rVert_2$ | 2.348 | 1.618 | 14.071 |
| | $\sum_i \sup\lvert\boldsymbol{z}_i - \hat{\boldsymbol{z}}_i\rvert/N$ | 0.006 | 0.006 | 0.022 |
| brca-3d | Objective Value | 1566.49 | **1566.40** | 1578.49 |
| | Best Bound | - | $-2.7 \times 10^4$ | **332.3** |
| | $\sup\lvert\boldsymbol{\pi} - \hat{\boldsymbol{\pi}}\rvert$ | 0.167 | 0.169 | 0.122 |
| | $\lVert\boldsymbol{\mu} - \hat{\boldsymbol{\mu}}\rVert_2$ | 394.07 | 401.47 | 418.05 |
| | $\sum_i \sup\lvert\boldsymbol{z}_i - \hat{\boldsymbol{z}}_i\rvert/N$ | 0.169 | 0.169 | 0.174 |

Table 2.4: Computational results of three approaches on three datasets.

We report best solution found and best bound. The total variation distance metric is used for $\boldsymbol{\pi}$ and $\boldsymbol{z}$, and the $L$-2 distance metric is used for $\boldsymbol{\mu}$. Based on the computational results in Table 2.4, MINLO performs best in terms of minimizing loss. MIQO provides tighter bound on the globally optimal value. While our goal is to find global optimality upper and lower bounds, the parameter estimation of MINLO and MIQO are roughly equivalent to EM. As presented in [6], Figure 2.6a shows that MIQO finds solution quickly, most of the remaining runtime is spent to prove global optimality of the solutions. While Figure 2.6b shows that as the sample size gets larger, runtime of MIQO improves comparing to MINLO. This shows that our MIQO is more promising than MINLO for larger data size. Nonetheless, more improvement on its computational performance to prove optimality is needed.




(a) Global convergence of MIQO.




(b) Comparison of MINLO and MIQO on computational runtime and sample size.

Figure 2.6: Comparison of MINLO and MIQO [6]

## 2.4 Conclusions and Future Work

We formulate MINLO for solving MAP. Our mixed-integer optimization model is easy to interpret statistically and adaptable to various prior distributions. We further reformulate MINLO to MIQO using PWL approximation and McCormick relaxations. We explore our customized B&B using the most-integral branching strategy although the performance is inferior to commercial solvers. The solution quality of MIQO is slightly better than EM and MINLO and is more promising when sample size is large [6].

Our MINLO and MIQO formulations can reasonably handle small datasets including around 50 observations. While the number of 50 sample size seems small, our methods can be useful in most biological experiments and clinical trials which study on small sample size. For larger datasets, our methods can provide upper and lower bounds of the solution. However, further study on the reformulation will certainly improve computational efficiency of this problem, one example is [7].

# Chapter 3

# Service Classification and Assignment of IT Tickets via Machine Learning and Optimization

## Introduction

We study mixed-integer optimization (MIO) for assigning the right tasks to the right workers. Our study creates a decision support system (DSS) to optimize the service classification and assignment of tickets using machine learning and MIO techniques for Information Technology Services (ITS) organizations. Typical ITS organization hires technicians to support users with a variety of incident or request services. The process begins with users submit a ticket describing an incident or request, then technicians receive and process tickets to fulfill a service. There are many levels of technicians who work on tickets. The process to fulfill a ticket either ends after a technician receives and fulfills a ticket or requires further communication among staff to help with processing the ticket. ITSM wants to avoid when tickets need further support from higher-tier technicians as there is an increasing cost for an organization when tickets need further support from higher-tier technicians.

Existing ticket assignment in general is in First-In-First-Out manner (FIFO) [8]. Upon receiving a ticket, a tier-1 technician must fully document a list of information, including incoming communication, summary and description, classification of the ticket according to the service catalog, and other meta-data attributes. Once documented, a ticket can go down one of three different routes. First, it can be escalated and assigned to another higher-tier technician. Second, it can be fulfilled directly by the technician. Third, it requires the tier-1 technician to consult with peers or manager first to gain consensus on how to process ticket. Figure 3.1 illustrates the current process to fulfill a ticket in the

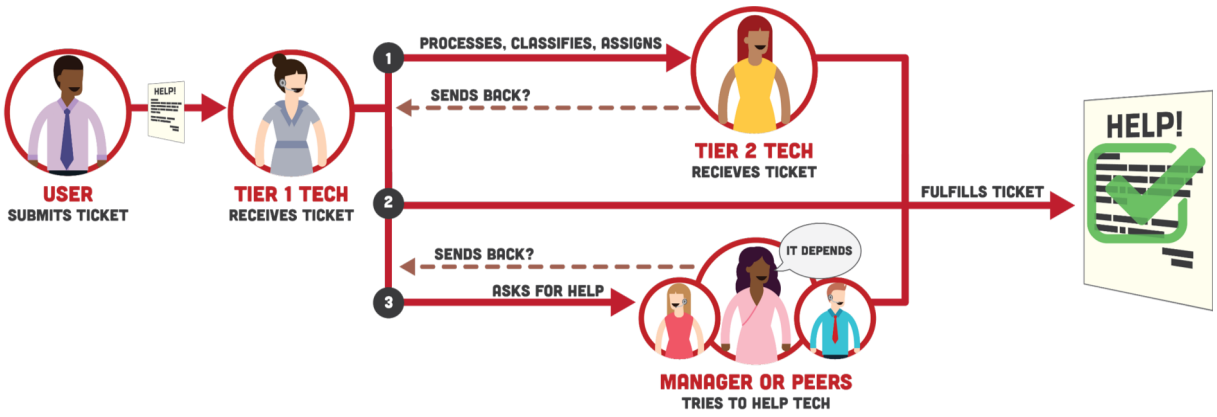



Figure 3.1: Current ITSM incident request process [9].

ITS management (ITSM).

There are many problems with the current ITSM process [9] that lead to extended processing time, increasing cost for organization, and inefficient ticket handling. New technicians require extensive and costly training about the service catalog and organization structure, advance training that is not practical for the job not only causes difficulty in ticket fulfillment but also increase training cost for ITS organization. A lack of knowledge management in a team or an individual can lead to ticket escalations that involve multiple technicians or managers to fulfill the ticket. Another problem is when technicians favor some types of tickets, e.g. easy tasks, lower-skilled or less specialized tier-1 technicians can be left with no tickets that they could work. A large size of staff can lead to inconsistency in handling tickets. Additionally, the lack of using historic data on incoming tickets is wasteful for the ticket process.

Total cost of a ticket is referred to as total cost of ownership (TCO). TCO of a single ticket includes the cost of support from all sources, including the service desk, desktop support, other groups or individuals in IT, and vendors [10]. Estimations of TCO on a per ticket basis is a compounding cost for each escalation as shown in Figure 3.2 and Table 3.1.

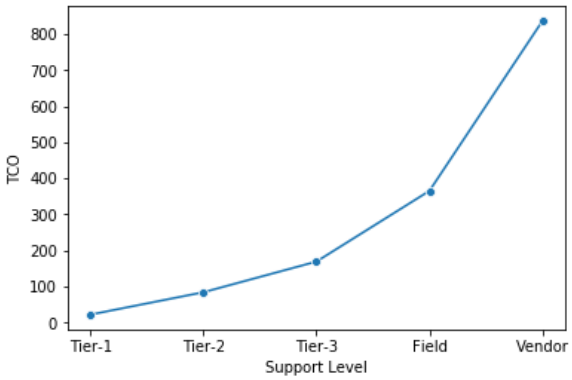



Figure 3.2: Accumulated total cost at each escalation

| Support Level | Cost Per Ticket | TCO |
|---|---|---|
| Tier-1 | $22 | $22 |
| Tier-2 | $62 | $84 |
| Tier-3 | $85 | $169 |
| Field | $196 | $365 |
| Vendor | $471 | $836 |

Table 3.1: Total cost of ownership [10]

We propose a data-driven decision support system that makes use of machine learning and integer optimization to provide recommendations to technicians. We next describe our system in detail.

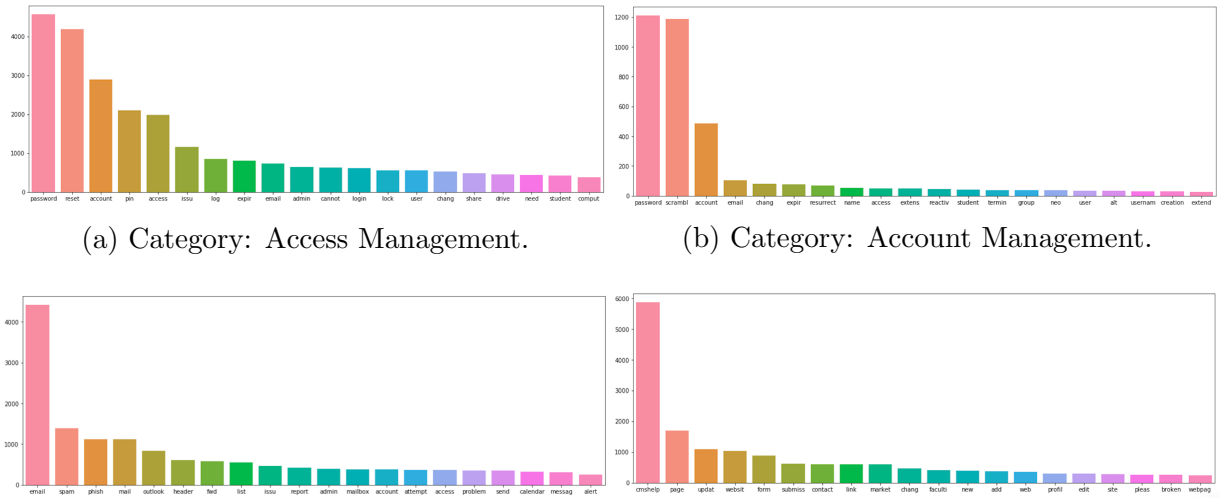
## 3.1 Methodology

We create a decision support system to address existing problems in typical IT services. Our proposed system consists of three stages: (1) extract keywords from an incoming ticket using natural language processing (NLP), (2) create a ticket classification model using machine learning techniques, and (3) create a ticket assignment model using integer optimization, as shown in Figure 3.3. The advantage of our system is that we make sure of optimization techniques, which have not been used extensively in the ITSM process and decision support systems [9].


Figure 3.3: An overview of our DSS process [9].

### 3.1.1 Extracting Keywords From Tickets Using NLP

We build our ticket classification models based on keywords extracted from both incoming tickets and a historic dataset. We clean the tickets before extracting, then create a dataset out of tickets using natural language processing (NLP).




(a) Category: Access Management.

(b) Category: Account Management.

(c) Category: Email.

(d) Category: Marketing Systems.

Figure 3.4: Keywords extracted from tickets using TF-IDF.

Feature extraction has two main methods: bag-of-words (BoW) and word embedding. BoW is based on word frequency, whereas word embedding is based on vector space model. While word embedding outperforms BoW in the state-of-the-art, our initial work extracts keywords from tickets using BoW due to the domain specific of IT service terminology; later on, we apply machine learning techniques on the extracted keywords to build a classification model. There are two approaches to extract important features from texts using BoW by using word frequency and term frequency-inverse document frequency (TF-IDF). Although TF-IDF takes into account words that occur frequently, but carry less meaningful information; e.g. is, and, the, word frequency works better for our cleaned data. Figure 3.4 illustrates keywords extracted using TF-IDF method from four categories.

### 3.1.2 Building Ticket Classification Model

The automatic ticket classification can save time and reduce workload of technicians. After extracting keywords, we build classification models using machine learning techniques to map tickets to service catalog. A service catalog contains a collection of all services that an IT Service organization provides to its users. The target of the classification can be either Category, SubCategory, or the combination of Service, Category, and SubCategory. We select Category as the target since it is easiest to deal with the number of classes and class imbalance. For the input, we use Summary and Description of tickets. Although ticket Summary displays the type of services better than ticket Description, some ticket Summary are missing. We create a main input variable that use Summary as an input if it is available, use Description otherwise. We truncate ticket categories in the service catalog that appeared less than 0.5% of the data for better prediction. We partition 70%

of data into training set and the remaining into test set.

We consider the following classification models: Naive Bayes (NB), Random Forest (RF), Logistic Regression (LR), Support Vector Machine (SVM), and Neural Network (NN). Our best model obtained from LR and NN with 78% accuracy. We summarize the results of our models in Table 3.2.

| | NB | LR | SVM | RF | NN |
|---|---|---|---|---|---|
| Frequency | 0.7398 | **0.7709** | 0.7468 | 0.7533 | **0.7721** |
| TF-IDF | 0.6987 | 0.6987 | 0.7322 | 0.7589 | 0.7685 |

Table 3.2: Accuracy results of each machine learning model using word frequency and TF-IDF.

### 3.1.3 Building Ticket Assignment Optimization Model

After classifying tickets, we build a ticket assignment model using mixed-integer optimization to assign tickets (tasks) technicians. The assignment considers required skills and time to fulfill ticket, skillset of technicians, cost, and availability of technicians. Given technician availability and types of tickets, the objective is to assign task to technician according to fit, while prioritizing balance between staffing levels. We minimize the underassignment and overassignment task time of technician by introducing penalty weight to the objective function. Penalty weight of underassignment task time is reciprocal to the rank, while penalty weight of overassignment task time is twice the penalty of the underassignment to distribute tasks among tier-1 technicians as equally as possible and allocate some availability to higher-tier technicians. We describe the assignment optimization model formulation in formulation (3.1) and notations used in Table 3.3.

$$\text{minimize} \quad \sum_{t\in T}\sum_{p\in P}\sum_{k\in K}|v_{pk}-v_{tk}|x_{tp}+\overline{w}_p\overline{Y}_p^2+\underline{w}_p\underline{Y}_p^2 \tag{3.1a}$$

$$\text{such that} \quad \sum_{p\in P} x_{tp}=1 \quad \forall\, t\in T, \tag{3.1b}$$

$$\sum_{t\in T} d_t x_{tp}\geq l_p \quad \forall\, p\in P, \tag{3.1c}$$

$$\sum_{t\in T} d_t x_{tp}-\overline{Y}_p+\underline{Y}_p=a_p \quad \forall\, p\in P, \tag{3.1d}$$

$$x_{tp}\in\{0,1\} \quad \forall\, t\in T,\ \forall\, p\in P, \quad \overline{Y}_p,\underline{Y}_p\geq 0 \quad \forall\, p\in P. \tag{3.1e}$$

| Sets | |
|---|---|
| $\mathcal{T}$ | Set of tasks, indexed by $t \in \mathcal{T}$ |
| $\mathcal{P}$ | Set of people (technicians), indexed by $p \in \mathcal{P}$ |
| $\mathcal{K}$ | Set of skills, indexed by $k \in \mathcal{K}$ |
| **Parameters** | |
| $a_p$ | Availability of person $p$, in hours |
| $l_p$ | Minimum working time required for person $p$ |
| $d_t$ | Time to complete task $t$ |
| $v_{tk}$ | Skill level requirement of skill $k$ for task $t$ |
| $v_{pk}$ | Skill level performance of skill $k$ for person $p$ |
| $\underline{w}_p$ | Penalty weight of underassignment 1-hour unit of work to person $p$ |
| $\overline{w}_p$ | Penalty weight of overassignment 1-hour unit of work to person $p$ |
| **Decision Variables** | |
| $x_{tp}$ | Binary variable assuming a value of 1 if task $t$ is assigned to person $p$; 0 otherwise |
| $\overline{Y}_p$ | Integer value of overassignment working time beyond the availability of person $p$, in hours |
| $\underline{Y}_p$ | Integer value of underassignment working time below the availability of person $p$, in hours |

Table 3.3: Notation for ticket assignment optimization model.

Objective function (3.1a) minimizes total absolute skill level difference of technician and task and weighted squared penalty for assignment time that undershoot/overshoot technician availability time. Constraint set (3.1b) ensures that a ticket is assigned to exactly one technician. Constraint set (3.1c) specifies the minimum working time required for each technician. Constraint set (3.1d) defines availability of each technician. Lastly, non-negativity and binary restrictions are defined in (3.1e). We next present a case study of our formulation at WPI Helpdesk Service.

## 3.2 A Case Study and Computational Experiments

We develop an ITSM tool and deploy to assign tickets to technicians at WPI Helpdesk Service. The experiment compares optimized (OPT) versus First-In-First-Out (FIFO) assignment on ticket throughput, timing, and quality of experienced and inexperienced technicians. The plan is to conduct the experiment daily on school days for six weeks as shown in Figure 3.5. At the start of each day, the manager of tier-1 service desk technicians initiates the ITSM Process, then technicians select a "give me work button" that provides OPT or FIFO tickets depending on the current treatment. Data has been collecting, with results to follow. Our system classifies tickets correctly and consistently 78% of the time, better than the classification done by typical tier-1 technicians. The system assigns various amount of tickets to technicians with balance workload and the

model is adjustable in many situations.

| | Day of Week | | | | |
|---|---|---|---|---|---|
| Week # | Monday | Tuesday | Wednesday | Thursday | Friday |
| 1 | OPT | FIFO | OPT | FIFO | OPT |
| 2 | FIFO | OPT | FIFO | OPT | FIFO |
| 3 | OPT | FIFO | OPT | FIFO | OPT |
| 4 | FIFO | OPT | FIFO | OPT | FIFO |
| 5 | OPT | FIFO | OPT | FIFO | OPT |
| 6 | FIFO | OPT | FIFO | OPT | FIFO |

Figure 3.5: Experimental design.

An example on a subset of WPI dataset shown in Table 3.4, our system spends roughly half a second to assign 57 tickets to eight technicians given working time of three hours each. Each ticket takes 0.84 hour to fulfill on average. For each technician, we report the average skill set value (AvgSkillSetValue), number of assignment tickets (NumTicketAssign), total working time (TotalWorkTime), overassignment time (OverAssignTime), and underassignment time (UnderAssignTime). Technicians who started working prior 2014 are considered higher-tier worker. We observe that there are three levels of technicians according to the AvgSkillSetValue, ranging between $< 2$, $[2, 3]$, and $> 3$. Our system distributes all tickets in such a way that lower-level technicians works more than higher-level technicians to give higher-level technicians more availability to supervise the IT process. Tickets are distributed to technicians in the same tier in the same amount as much as possible, except when some technicians do not possess required skill set to work on tickets. Table 3.5 displays the average results of the sample data.

| Technician ID | Start Date | AvgSkillSetVal | NumAssignTickets | AvgTimePerTicket | OverAssignTime | UnderAssignTime |
|---|---|---|---|---|---|---|
| 1 | 9/15/16 | 1.71 | 9 | 6.4 | 3.4 | 0 |
| 2 | 9/14/17 | 1.88 | 12 | 7.66 | 4.66 | 0 |
| 3 | 1/13/18 | 1.90 | 10 | 8.14 | 5.14 | 0 |
| 4 | 10/1/14 | 2.52 | 7 | 4.97 | 1.97 | 0 |
| 5 | 9/21/14 | 2.67 | 5 | 4.32 | 1.32 | 0 |
| 6 | 11/1/16 | 2.85 | 7 | 7.03 | 4.03 | 0 |
| 7 | 9/15/15 | 3.62 | 4 | 5.55 | 2.55 | 0 |
| 8 | 8/25/12 | 3.69 | 3 | 3.61 | 0.61 | 0 |

Table 3.4: Ticket assignment results from a sample dataset.

## 3.3 Conclusions

We create an automatic system to process tickets in IT services. Our system consists of two main features: ticket classification and assignment. Through the use of machine learning and integer optimization, we hope to conserve resources and increase efficiency

| AvgSkillSetValRange | Average | | |
|---|---|---|---|
| | NumAssignTickets | TotalWorkTime | OverAssignTime |
| $\leq 2$ | 10.33 | 7.40 | 4.40 |
| $(2, 3]$ | 6.33 | 5.44 | 2.44 |
| $> 3$ | 3.50 | 4.58 | 1.58 |

Table 3.5: Averaging results of a sample dataset.

in ticket handling. Our systems precision can be increased over time as the training set gains more tickets and the system matures. Additionally, the system is adaptable to other applications; for example, an auto repair application where issues are classified to a category in a repair catalog (OBD – On-Board-Diagnostic trouble codes) then assigned to appropriate mechanics, or an medical application where symptoms are classified to potential diagnosis (ICD-9/10 – International Classification of Diseases) then assigned to appropriate medical staff.

# Chapter 4

# A Comparative Study of Stability Representations for Solving Many-To-One Matching Problems with Ties and Incomplete Lists via Integer Optimization

## Introduction

We study integer optimization for many-to-one matchings with incomplete preference lists and ties. Many-to-one matching appears in a variety of applications, such as in school choice [11–13], college application [14–16], hospital-residency matching [17, 18], and refugee resettlement [19, 20]. Multiple algorithmic approaches such as deferred acceptance (DA) and top trading cycle (TTC) algorithms are widely used in many-to-one matching; however, optimization-based approaches offer distinct advantages over traditional algorithmic approaches, in their ability to accommodate side constraints and readily handle incomplete preference lists with ties. We capitalize on this within an integer optimization framework for many-to-one matching. Our many-to-one matching problem is student-to-project-center assignment (SPC) where each student is assigned to a project center or unmatched, project centers have a limited capacity, and the assignment considers preference of both sides. SPC assignment may further require the creation of a cohort, which is a group of students having one or more particular features in common, such as academic major or gender.

Well-designed matching mechanisms feature efficiency, stability (fairness), and strategy-

proofness properties. We define an entity to be either a student or a project center director. In term of efficiency, a matching is *Pareto efficient* if there exists no other match where an entity is better off without making at least one other entity worse off. A *blocking pair* is a pair of student $s$ and project center $p$ in which both student $s$ and project center $p$ prefer one another to their actual assignments or their unassigned (undersubscribed) status. A matching has *justified envy* if there exists a blocking pair $(s, p)$ in which student $s$ prefers project center $p$ to the current match and, concurrently, project center $p$ prefers student $s$ over (at least) one other students in its current assignment. A matching has *waste* if there exists a blocking pair $(s, p)$ in which student $s$ prefers project center $p$ to the current match and, concurrently, project center $p$ has an empty seat and also prefers student $s$ over a vacant position. A matching is *stable* if there exists no blocking pair characterizing either justified envy or waste. A mechanism is *strategy-proof* if it is not possible for an entity to obtain a better outcome by misreporting their true preference. While all three properties may be desirable, [21] show that there exists no mechanism with all three properties simultaneously.

Multiple approaches for solving many-to-one matching problem offer different desirable properties suiting the particular context. An outcome of the DA algorithm is proved to be stable and efficient for the proposing side [14]. Whereas, an outcome of the TTC algorithm is proved to be strategy-proof and efficient [11]. If fairness is preferred to efficiency, then one should select DA algorithm; and vice versa [11]. On the other hand, optimization-based approaches can be designed to not only maximize efficiency but also ensure stable outcomes by adding constraints [see, e.g., 18, 22–24]. Unlike aforementioned approaches, optimization-based approaches can readily accommodate side constraints and incomplete preference lists with ties.

We present an integer optimization model for many-to-one matchings with incomplete preference lists and ties. Our objective functions impose a strict lexicographical ordering on the priority of individual components. We introduce multiple new many-to-one blocking pair elimination constraints that ensure stable match outcomes. We further create algorithms to accelerate the computational performance of our stability constraints. We incorporate the concept of cohorts into the model via a goal programming framework that penalizes deviations in a quadratic manner. Lastly, we conduct comprehensive experiments to study the computational performance of our stability constraints and compare them with constraints from the extant literature, under a variety of conditions. Our experiments on both real and synthetic data reveal the model properties where each method excels, and also show that our stability constraints are more computationally efficient than the existing methods for typical real-world matching applications where sufficient seats exist for applicants, such as school choice problems and hospital-residency matching.

## 4.1 Background

The Deferred Acceptance algorithm (DA) was originally designed for the one-to-one stable marriage problem [14] and, later, extended to the many-to-one college admission problem [25]. The algorithm is an iterative process in which each of the entities on one side proposes to entities on the other side. An outcome of the DA algorithm for the stable marriage problem was shown to be stable and efficient for the proposing side [14]. Some properties of the DA algorithm applied to one-to-one matchings do not carry over to many-to-one matchings. The author of [25] shows that no stable matching mechanism is strategy-proof for the project center side, but student-proposing DA is strategy-proof for students. The author of [26] shows that project centers may benefit by misreporting capacities, and that no stable matching outcome is non-manipulable via capacities. In fact, [27] shows that there exists no stable matching mechanism that always yields an *incentive compatible* outcome in which both sides of the entities give their truthful preferences.

Another popular approach for many-to-one matching is the TTC [28]. The TTC algorithm starts with unassigned students and available schools. Each student points to her best school, and each school points to the student with the highest priority. Then, there must be at least one trading cycle, that is, an ordered list of students and schools in which the first student in the list points to the first school in the list (her best school) and the first school of the list points to the second student of the list (the student with the highest priority) and so on. Each student in a cycle is assigned to the school she points to and is removed. The capacity of each school reduces by one, and if it reaches zero, the school is removed. The algorithm terminates when all students are assigned or all submitted choices are considered. [11] explain TTC for school choice and show that TTC is strategy-proof and efficient but does not always yield stable outcomes. Both DA and TTC require strict preference ordering, that is, ties among preferences will render the algorithms inapplicable; thus, ambivalence can be accommodated only by breaking such ties with small random perturbations. Depending on the application, if stability is favored over full efficiency, then DA is preferable — as it yields the most (proposing side) efficient outcomes that ensure stability; whereas if efficiency is favored over stability, then TTC is preferable to DA [11].

In contrast to standard algorithmic approaches such as DA and TTC, we use techniques from integer optimization (IO) to develop approaches for determining optimal many-to-one matching outcomes. Table 4.1 summarizes studies that present optimization-based models in the context of stable matching, using various approaches to model the concept of stability.

Vande Vate [29] is among the first to use optimization to solve the one-to-one stable marriage problem. The study introduces a particular form of constraints that ensure the

| Authors (year) | Type of Matchings | Optimization Objective | Side Constraints |
|---|---|---|---|
| Vande Vate (1989) | One-to-one | max social values of marriage problem | stability constraints for one-to-one matching describing the stable marriage polytope |
| Roth et al. (1993) | One-to-one | max utility of marriage problem | stability constraints for one-to-one matching |
| Baïou and Balinski (2000) | Many-to-one | max utility associated with the assignment | stability constraints for many-to-one matching describing the stable admissions polytope |
| Kwanashie and Manlove (2014) | Many-to-one | max number of matches | constraint encodings for stable matching |
| Ágoston et al. (2016) | Many-to-one | min total score-limits or free objective function | score-limits, lower and common quotas, and paired applications |
| Delorme et al. (2019) | Many-to-one | max number of matches | stability constraints representing in different models |
| Shimada et al. (2020) | Many-to-one | max total utility; min total blocking pairs; min total squared shortages of subgroup members from target | stability constraints and constraints for matching in groups |
| *Current study* | Many-to-one | max number of matches; max student utility; max project center utility; min total squared deviation from target cohort | stability constraints and constraints for cohort construction |

Table 4.1: Optimization-based approaches in the literature that consider stability.

matching outcomes are stable, and moreover it is shown that they preserve the integrality of the variables representing the assignment of man $i$ to woman $j$, thus enabling solution via computationally efficient linear programming techniques. Roth et al. [30] explore the dual and fractional solutions of the stable marriage linear program from [29]. They show that the dual of this particular linear program has a strong relationship to the primal, namely, that each optimal solution to the primal is contained in an optimal solution to the dual, highlighting the duality structure of the underlying linear program.

Baïou and Balinski [31] are the first to present stability constraints for many-to-one matching. They describe the stable admission polytope — the convex hull or the smallest convex set of the stable assignments of the college admissions problem. Kwanashie and Manlove [32] present a set of stability constraints for the Scottish hospital-residency matching with a computational study on their models using both randomly generated and real-world datasets. The form of their presented stability stability constraints also appears in their recent work in [18]. Ágoston et al. [23] focus on the Hungarian admissions problem with four special features: the solution concept of stable score-limits, the presence of lower and common quotas, and paired applications. The score-limit computed for each college is the lowest score that allows students to be admitted. The stable score-limits ensure stable outcomes resulted from using score-limits, in which students are admitted to their top choice where their score achieved the score limit. Similar to quotas in school choice, lower quotas define the minimum number of assigned students for each college to remain open and upper quotas limit the maximum number of students that can be assigned to each college. The idea of paired applications is similar to many-to-one matching with couples.

Delorme et al. [18] present integer linear programming (ILP) models for one-to-one pairing of children with adoptive families and many-to-one hospital-residency assignment, each of which maximizes the number of stable matches. They also derive an enhanced model that reduces the number of nonzero elements with respective forms of stability

constraints and conduct experiments on real and random datasets. Shimada et al. [24] also consider many-to-one matching for hospital-residency. They address the trade-off between stability and efficiency through the use of multiple objectives. Their multi-objective optimization model weights the maximum total utility, the minimum number of blocking pairs, and the minimum small-subgroup penalty — the squared shortages of subgroup members deviating from the target number for the purpose of matching in groups (cohorts).

In many real-life applications, it is unrealistic to require preferences over every entity on the other side, and moreover it may be the case that an entity has ambivalence in their preferences over multiple entities. Optimization-based approaches excel in such contexts of matching with incomplete lists and ties among preferences. Our study solves the more general problem that allows for weighted matches in the context of incomplete lists and ties. In particular, students can have ties over project centers, while project centers can also have ties among students (though rarely do).

It is often desirable in school choice problems and residency applications to have a matching outcome with an applicant blend such as cultural and racial diversity. In the context of school choice this has been studied as a minimum/maximum quota problem, where students are divided into several subgroups according to their socioeconomic status, and each school imposes a minimum/maximum quota for each group of students. Hogan [33] introduced the concept of matching in groups, which focuses on using the DA algorithm for many-to-one matching with stability. The use of optimization for matching couples, roommates, or people with similar backgrounds has also seen recent investigation [see e.g., 23, 24, 34–37]. In our SPC assignment, project center directors may consider cohort, or group, formation of students. Some project centers may seek a minimum number of female students to balance subsequent team formation, or reduce dormitory costs, whereas others may request a specific skill from a list of student majors.

## 4.2 Model Formulations

We present a lexicographical multi-objective optimization formulation for the SPC matching problem that considers both the total number of matches and the total market utility, where maximizing the total number of matches is the most important objective. Eligible assignments consist of students to project centers that each rank one another on their preference lists.

Define $\mathcal{S}$ to be a set of students and $\mathcal{P}$ to be a set of project centers. Some students may be unmatched due to incomplete list, we augment the set of project centers $\mathcal{P}$ to include a virtual project center $p_0$ dedicated to unmatched students. We assume that it

is preferable for a project center to accept any eligible student over a vacant seat.

Define $q_{sp}$ to be student preference value and $k_{sp}$ to be project center director preference value of student $s$ at project center $p$, respectively. Define $r_{sp}$ to be integer ordinal preference value of student $s$ at project center $p$, where smaller values of $r_{sp}$ indicate a more preferable project center for student $s$. The preference order $\succ$ indicates an entity in the left-hand side is more preferable with a corresponding greater utility value than the right-hand side. Define the value of assigning student $s$ to project center $p$, also known as the market utility, as $u_{sp} := U(q_{sp}, k_{sp})$, which is a linear function of cardinal preferences of student $s$ and project center $p$.

Define $\mathcal{P}_s$ to be the set of project centers that student $s$ ranks. Students prefer being unmatched to being assigned to undesirable project centers that are not in their list, that is, $q_{sp} > q_{sp_0} > 0 \ \forall \ s \in \mathcal{S}, \ \forall \ p \in \mathcal{P}_s$. Moreover, a virtual project center prefers all students equally, $k_{sp_0} = 0 \ \forall \ s \in \mathcal{S}$, and assigning students to a virtual project center does not contribute to the objective function, that is, $u_{sp_0} = 0 \ \forall s \in \mathcal{S}$. We present the baseline optimization formulation for SPC matching problem in formulation 4.1 and notation used in the formulation in Table 4.2

| Sets | |
|---|---|
| $\mathcal{S}$ | Set of students, indexed by $s$ |
| $\mathcal{P}$ | Set of project centers including a virtual project center $p_0$, indexed by $p$ |
| $\mathcal{R}_s$ | Set of ranking levels of student $s$ over the project center preference list, indexed by $r \leq \lvert\mathcal{S}\rvert$ |
| $\mathcal{R}_p$ | Set of ranking levels of project center $p$ over the student preference list, indexed by $r \leq \lvert\mathcal{P}\rvert$ |
| $\mathcal{S}_p$ | Set of students for whom project center $p$ ranks in its preference list, indexed by $s$ |
| $\mathcal{P}_s$ | Set of project centers that student $s$ ranks in their preference list, indexed by $p$ |
| $\mathcal{S}_p^{(r)}$ | Set of students for whom project center $p$ ranks in level $r \in \mathcal{R}_s$, indexed by $s^{(r)}$ |
| $\mathcal{P}_s^{(r)}$ | Set of project centers that student $s$ ranks in level $r \in \mathcal{R}_p$, indexed by $p^{(r)}$ |
| **Parameters** | |
| $u_{sp}$ | Utility value of matching student $s$ to project center $p$ |
| $q_{sp}$ | Student preference of student $s$ at project center $p$ |
| $k_{sp}$ | Director preference of student $s$ at project center $p$ |
| $r_{sp}$ | Integer ordinal preference value of student $s$ for project center $p$ |
| $c_p$ | Capacity of project center $p$ |
| **Decision Variables** | |
| $x_{sp}$ | Binary variable assuming a value of 1 if student $s$ is assigned to project center $p$; 0 otherwise |
| $z_p$ | Indicator with value of 1 if project center $p$ has at least one empty seat |
| $\boldsymbol{x}$ | Entire $\lvert\mathcal{S}\rvert \times \lvert\mathcal{P}\rvert$ match outcome |

Table 4.2: Notation for matching student to project center models.

$$\text{maximize} \quad \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} x_{sp} + \gamma \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} u_{sp}x_{sp} \tag{4.1a}$$

$$\text{subject to} \quad \sum_{p\in\mathcal{P}} x_{sp} = 1 \quad \forall \ s \in \mathcal{S}, \tag{4.1b}$$

$$\sum_{s \in \mathcal{S}} x_{sp} \leq c_p \quad \forall\ p \in \mathcal{P}, \tag{4.1c}$$

$$x_{sp} = 0 \quad \forall\ s \in \mathcal{S},\ \forall\ p \in \mathcal{P}_s : p_0 \succ_s p, \tag{4.1d}$$

$$[\ \textit{Blocking pair elimination constraints}\ ], \tag{4.1e}$$

$$x_{sp} \in \{0, 1\} \quad \forall\ s \in \mathcal{S},\ \forall\ p \in \mathcal{P}. \tag{4.1f}$$

Objective function (4.1a) maximizes the sum of total student placement and weighted total market utility via the careful selection of coefficient $\gamma$ that strictly prioritizes the maximization of total student placement.[1] Constraint set (4.1b) requires that each student must be assigned to exactly one project center, whether a project center in the preference list, or the virtual project center $p_0$ introduced for unmatched students. Constraint set (4.1c) ensures that the number of students assigned to each project center does not exceed site capacity. Constraint sets (4.1d) and (4.1e) together ensure stability; constraint set (4.1d) preserves individual rationality, while constraint set (4.1e) prohibits solutions with blocking pairs. In Section 4.3 we introduce several new many-to-one stability constraint representations that prevent blocking pairs so as to ensure stable match outcomes, and discuss their relationship to existing forms in the literature. Variable domains are stated in (4.1f). We will extend this baseline formulation to create other models through our study.

## 4.3 New and Existing Linear Representations of Stability

Optimization formulation (4.1) maintains strict conditions of stability through constraints that express the prevention of blocking pairs. Recall that a match is *stable* if there exists no blocking pair characterizing justified envy or waste. This definition has already been represented in several mathematical forms for both one-to-one [see, e.g., 29, 30] and many-to-one matching markets in the literature [see, e.g., 18, 31]. We will first review the existing stability constraints in the literature, and then present new and alternative linear forms of stability that eliminate justified envy and waste, respectively.

### 4.3.1 Existing Stability Constraints

The author of [29] introduces a binary integer optimization formulation that appears to be the first to model stability in the context of the one-to-one stable marriage problem, which

[1] We set coefficient $\gamma$ equal to the reciprocal of total available capacity at project centers plus a small positive value to ensure that the weighted total market utility is strictly less than one, thereby ensuring any single additional placement is preferred over the greatest market utility.

assumes $|\mathcal{M}| = |\mathcal{W}|$ and complete preference lists with strict ordering over all candidates from on other side. This study also shows that the optimal solution to the linear relaxation of the presented integer program that models the one-to-one stable marriage problem has integral extreme points. Denote $\mathcal{M}$ to be a set of men, $\mathcal{W}$ a set of women, and $x_{mw} = 1$ if man $m$ is matched to woman $w$; 0 otherwise. This constraint set is presented in (VV), which imposes the condition of stability by requiring that if woman $w$ marries someone less desirable than man $m$, then $m$ must marry someone more desirable than $w$:

$$\sum_{i \prec_w m} x_{iw} - \sum_{j \succ_m w} x_{mj} \leq 0 \quad \forall\ m \in \mathcal{M},\ \forall\ w \in \mathcal{W}. \tag{VV}$$

The authors of [31] appear to be the first to use a set of linear constraints to express the concept of stability for many-to-one matching in the context of complete preference lists and strict ordering of preferences. This is presented in constraint set (BB), which imposes stability by enforcing that if student $s$ is not assigned to project center $p$ ($x_{sp} = 0$), then either student $s$ is assigned to project center $j$ that is preferred to $p$, or all of the $c_p$ seats of project center $p$ are assigned to students that $p$ prefers to $s$:

$$c_p x_{sp} + c_p \sum_{j \succ_s p} x_{sj} + \sum_{i \succ_p s} x_{ip} \geq c_p \quad \forall\ s \in \mathcal{S},\ \forall\ p \in \mathcal{P}. \tag{BB}$$

Adapted to the context with ties, (BB) can be modified to (BBT) whereby stability is imposed by also considering the assignment of student $s$ to other project centers ranked identically to project center $p$ and the assignment of other students ranked identically to student $s$ to project center $p$, when $s$ is not assigned to $p$:

$$c_p x_{sp} + c_p \sum_{\substack{j \neq p: \\ j \succeq_s p}} x_{sj} + \sum_{\substack{i \neq s: \\ i \succeq_p s}} x_{ip} \geq c_p \quad \forall\ s \in \mathcal{S},\ \forall\ p \in \mathcal{P}. \tag{BBT}$$

This resembles the constraint set presented in [32] as well as [18], who present integer optimization frameworks that embed linear stability constraints in the context of many-to-one matching with ties and incomplete lists. Their constraint set presented as (HRT) below imposes the condition of stability by ensuring that if student $s$ is not assigned to project center $p$ or any project center that $s$ prefers at least as much as $p$ (that is, $\sum_{j \succeq_s p} x_{sj} = 0$), then project center $p$ must fill its capacity with students it ranks at the same level or higher than $s$:

$$c_p \left(1 - \sum_{j \succeq_s p} x_{sj}\right) \leq \sum_{i \succeq_p s} x_{ip} \quad \forall\ s \in \mathcal{S},\ \forall\ p \in \mathcal{P}. \tag{HRT}$$

We note the similarity between (BBT) and (HRT), as (HRT) can be derived from (BBT) when generalizing to ties and incomplete lists. Additionally, the form of (BBT) is also appropriate for incomplete lists. As limited computational testing reveals that both (BBT)

and (HRT) exhibit similar computational performance, we elect to proceed with (BBT) for the remainder of our discussion.

### 4.3.2 Proposed Stability Constraints

We introduce constraint sets that eliminate blocking pairs associated to justified envy and a system of constraints that eliminates blocking pairs associated to waste; thus, both together ensure stability of the match outcomes. We first extend the one-to-one stability constraint set of [29] to obtain a new stability constraint set for the many-to-one context; combining with our system that eliminates blocking pairs associated to waste, the many-to-one (VVM) sufficiently ensures stable outcomes. We then introduce a new pairwise form of linear constraints to prevent blocking pairs. We also derive two useful and related forms through aggregation, and further introduce two algorithms to enhance computational performance of their construction.

**Extending (VV) to Eliminate Justified Envy in the Many-to-one Context.**

We derive a many-to-one stability constraint from the one-to-one stability constraint (VV) in [29]. If we view men ($m$) and women ($w$) in the one-to-one stability constraint (VV) as students ($s$) and project centers ($p$), respectively, we can extend the concept of stable marriage of [29] in constraint set (VV) to the many-to-one context with ties among preferences and incomplete lists where students may be matched to the virtual project center as shown in (VVM).

$$\sum_{i \prec_p s} x_{ip} \leq c_p \left(1 - \sum_{j \prec_s p} x_{sj}\right) \quad \forall\ s \in \mathcal{S},\ \forall\ p \in \mathcal{P}. \tag{VVM}$$

Constraint set (VVM) eliminates blocking pair $(s, p)$ associated to justified envy by ensuring that if student $s$ is assigned to a project center less desirable than $p$, then project center $p$ cannot be assigned a student less preferable than $s$. This also implies that if project center $p$ has its capacity filled by students less desirable than $s$, then student $s$ cannot be assigned to a center less preferable than $p$.

**Introducing A System of Constraints to Ensure Outcomes Without Waste.**

A blocking pair $(i, p)$ associated to waste occurs when student $i$ prefers project center $p$ to their match and project center $p$ prefers student $i$ to an empty seat. We introduce a system of constraints that are sensitive to when project centers have empty seats, and restrict this type of blocking pair for our proposed stability constraint sets.

The number of empty seats at each project center is the difference between the capacity

and the number of students assigned to the center. Define $z_p$ as an indicator that takes value of 1 if project center $p$ has at least one empty seat ($c_p - \sum_{s \in \mathcal{S}} x_{sp} > 0$). We create the (NWS) system of constraints to prevent blocking pairs associated to waste:

$$\sum_{i \in \mathcal{S}_p} \sum_{\substack{j \in \mathcal{P}: \\ j \prec_i p}} x_{ij} \leq \mathcal{M}_w (1 - z_p) \quad \forall\, p \in \mathcal{P}, \tag{NWa}$$

$$z_p \leq c_p - \sum_{s \in \mathcal{S}} x_{sp} \leq c_p z_p \quad \forall\, p \in \mathcal{P}, \tag{NWb}$$

$$z_p \in \{0, 1\} \quad \forall\, p \in \mathcal{P}. \tag{NWc}$$

Constraint set (NWa) prevents blocking pair $(i, p)$ when waste exists by ensuring that if there exists an empty seat at project center $p$ ($z_p = 1$), then there exists no match $(i, j) \in \mathcal{S} \times \mathcal{P}$ in which project center $p$ prefers student $i$ to an empty seat and student $i$ prefers project center $p$ to project center $j$. We set the upper bound of the number of possible blocking pairs at project center $p$, $\mathcal{M}_w$, to be the number of students who rank project center $p$ in their preference list. Constraint set (NWb) ensures that $z_p = 1$ if there is at least one empty seat at project center $p$, that is, $c_p - \sum_{s \in \mathcal{S}} x_{sp} > 0$, and 0 otherwise. Variable domains are covered in (NWc).

The (NWS) system eliminates blocking pairs associated to waste. They complement (VVM) and all other stability constraint sets we introduce that eliminate blocking pairs associated to justified envy, thereby ensuring stability. However, (NWS) is superfluous and may be dropped when combined with an objective function that prioritizes student utility. This special objective function induces stability because prioritizing student utility incentivizes assigning students to their best available choice. As any desirable project centers prefer accepting students in their list to having empty seats, the outcomes will have no empty seats that students would value over their own assignments and, thus, result in stable match outcomes.

**Introducing New Stability Representations To Eliminate Justified Envy.**

We now introduce a new linear representation of stability that, similar to (VVM), forbids blocking pairs associated to justified envy. We then show how to derive two new linear representations of stability via aggregation.

The notion of justified envy in stable markets states that pairs $(s, p)$ and $(i, j)$ cannot be simultaneously matched if either of the blocking pairs $(s, j)$ or $(i, p)$ exist in which their members prefer one other to the actual match. In other words, if student $s$ prefers project center $j$ to $p$ and project center $j$ also prefers student $s$ to $i$, then matching pairs $(s, p)$ and $(i, j)$ would have been unstable. Similarly, if student $i$ prefers project center $p$ to $j$ and project center $p$ prefers student $i$ to $s$, then the initial match would have been

unstable. This condition of stability that prevents any such blocking pairs is represented in the following pairwise elimination (PW) constraint set:

$$x_{sp} + x_{ij} \leq 1 \quad \forall\, s, i \in \mathcal{S},\ \forall\, p, j \in \mathcal{P} : ((j \succ_s p) \wedge (i \prec_j s)) \vee ((i \succ_p s) \wedge (j \prec_i p)). \quad \texttt{(PW)}$$

Although stability constraint set (PW) is intuitive, there are $\mathcal{O}(|\mathcal{S}|^2|\mathcal{P}|^2)$ constraints needed to ensure stability. In an effort to reduce the number of constraints, we propose two approaches to aggregate over $i$ and $j$ in (PW) that satisfy the *first* and *second* set of blocking pair conditions, then present each approach in constraint sets (SPC1) and (SPC2), respectively.

$$\sum_{\substack{j \in \mathcal{P}:\\ j \succ_s p}} \sum_{\substack{i \in \mathcal{S}:\\ i \prec_j s}} x_{ij} \leq \mathcal{M}_s(1 - x_{sp}) \quad \forall\, s \in \mathcal{S},\ \forall\, p \in \mathcal{P}. \tag{SPC1}$$

The constant $\mathcal{M}_s$ upper bounds the number of possible blocking pairs associated with student $s$. Following the first set of blocking pair conditions in (PW), constraint set (SPC1) ensures that if student $s$ is assigned to project center $p$, then no pair $(i, j)$ is matched where student $s$ prefers project center $j$ to $p$ and project center $j$ prefers student $s$ to $i$.

An alternative way to aggregate stability constraint set (PW) is

$$\sum_{\substack{i \in \mathcal{S}:\\ i \succ_p s}} \sum_{\substack{j \in \mathcal{P}:\\ j \prec_i p}} x_{ij} \leq \mathcal{M}_p(1 - x_{sp}) \quad \forall\, s \in \mathcal{S},\ \forall\, p \in \mathcal{P}. \tag{SPC2}$$

The constant $\mathcal{M}_p$ upper bounds the number of possible blocking pairs associated with project center $p$. Following the second set of blocking pair conditions in (PW), constraint set (SPC2) ensures that if student $s$ is assigned to project center $p$, then no pair $(i, j)$ is matched where student $s$ prefers project center $j$ to $p$ and project center $j$ prefers student $s$ to $i$.

Even with aggregation, the construction of constraint sets (SPC1) and (SPC2) can be time consuming, especially for large values of $|\mathcal{S}|$ and $|\mathcal{P}|$. Therefore, we propose a recurrence relation to reduce the number of variables and improve the construction time and computational performance of constraint sets (SPC1) and (SPC2) in Algorithm 1 and Algorithm 2, respectively.

**Enhancing the Construction and Performance of (SPC1) and (SPC2).**

Algorithm 1 generates stability constraint set (SPC1) and ensures that if student $s$ is assigned to project center $p^{(r)}$, then there exists no assignment of student $i$ to project center $p^{(k)}$ where student $s$ prefers more than $p^{(r)}$, and $p^{(k)}$ strictly prefers $s$ to $i$, that is, it prevents blocking pairs $(s, p^{(k)})$, $\forall\ k \in \{1, \ldots, r-1\}$. As a result, Algorithm 1 efficiently constructs constraints that prohibit blocking pairs associated with student $s$.

We next present Algorithm 2, which generates stability constraint set (SPC2) in a

**Algorithm 1:** Recurrence relation for constructing (SPC1)

**foreach** $s \in \mathcal{S}$ **do**
  Rank $p \in \mathcal{P}$: $p^{(r)} \in \mathcal{P}_s^{(r)}, q_{sp^{(1)}} > q_{sp^{(2)}} > \dots$ // Order project centers in preference list of student $s$ in nonincreasing fashion.
  **for** $r = 1$ **to** $|\mathcal{R}_p|$ **do**
    **foreach** $p^{(r)} \in \mathcal{P}_s^{(r)}$ **do**
      Define $\beta_{sp^{(r)}} = \sum_{i \in \mathcal{S}: i \prec_{p^{(r)}} s} x_{ip^{(r)}}$ // Sum over those students $i$ that project center $p^{(r)}$ prefers strictly less than $s$.
      **if** *r=1* **then**
        Define $\alpha_{sp^{(1)}} = 0$ // There exists no blocking pair if student $s$ is assigned to their first choice.
      **else**
        Define $\alpha_{sp^{(r)}} = \alpha_{sp^{(r-1)}} + \sum_{p \in \mathcal{P}_s^{(r-1)}} \beta_{sp}$ // If student $s$ is assigned to $p^{(r)}$, then $\alpha_{sp^{(r)}}$ enumerates through project centers ranked better than $r$ and prevents blocking pairs associated with $s$.
      Add a constraint: $\alpha_{sp^{(r)}} \leq \mathcal{M}_s(1 - x_{sp^{(r)}})$ // Add stability constraint (SPC1)

similar manner to Algorithm 1. Algorithm 2 ensures that if project center $p$ is assigned student $s^{(r)}$, then there exists no assignment of a project center $j$ to student $s^{(k)}$ who project center $p$ prefers more than $s^{(r)}$, and $s^{(k)}$ strictly prefers $p$ to $j$, that is, it prevents blocking pairs $(s^{(k)}, p)$, $\forall\, k \in \{1, \dots, r-1\}$. As a result, Algorithm 2 efficiently constructs constraints that prohibit blocking pairs associated with project center $p$.

**Algorithm 2:** Recurrence relation for constructing (SPC2)

**foreach** $p \in \mathcal{P}$ **do**
  Rank $s \in \mathcal{S}$: $s^{(r)} \in \mathcal{S}_p^{(r)}, k_{s^{(1)}p} > k_{s^{(2)}p} > \dots$ // Order students in preference list of project center $p$ in nonincreasing fashion.
  **for** $r = 1$ **to** $|\mathcal{R}_s|$ **do**
    **foreach** $s^{(r)} \in \mathcal{S}_p^{(r)}$ **do**
      Define $\beta_{s^{(r)}p} = \sum_{j \in \mathcal{P}: j \prec_{s^{(r)}} p} x_{s^{(r)}j}$ // Sum over those project centers $j$ that student $s^{(r)}$ prefers strictly less than $p$
      **if** $r = 1$ **then**
        Define $\alpha_{s^{(1)}p} = 0$ // There exists no blocking pair if project center $p$ is assigned its first choice.
      **else**
        Define $\alpha_{s^{(r)}p} = \alpha_{s^{(r-1)}p} + \sum_{s \in \mathcal{S}_p^{(r-1)}} \beta_{sp}$ // If center $p$ is assigned student rank $r$, then $\alpha_{s^{(r)}p}$ enumerates through students ranked better than $r$ and prevents blocking pairs associated with $p$.
      Add a constraint: $\alpha_{s^{(r)}p} \leq \mathcal{M}_p(1 - x_{s^{(r)}p})$ // Add stability constraint (SPC2)

The auxiliary variables $\alpha_{sp}$ and $\beta_{sp}$ $\forall\ s \in \mathcal{S}$, $\forall\ p \in \mathcal{P}$ in Algorithms 1 and 2 are defined as nonnegative integer variables and constructed over sorted preference lists. The

value of $\mathcal{M}_s$ is no more than $r(|\mathcal{S}|-1)$ and the value of $\mathcal{M}_p$ is no more than $r(|\mathcal{P}|-1)$; each is the count of the possible blocking pairs that are available from inspection of the (incomplete) preference lists of $s$ and $p$, respectively. We illustrate Algorithms 1 and 2 with a case in Figure 4.1 in Appendix C. Later in Section 4.5, we conduct comprehensive experiments to compare the computational performance of (`SPC1`), (`SPC2`), (`VVM`), and (`BBT`), and investigate the performance of these constraint sets under a variety setting of parameters.

$$
\begin{array}{ll}
s_1: \ (p_1) & p_1: \ (s_2, s_3), (s_1) \\
s_2: \ (p_1, p_2) & p_2: \ (s_1), (s_2), (s_3) \\
s_3: \ (p_2), (p_1) &
\end{array}
$$

Figure 4.1: The illustration of the SPC assignment with incomplete preference lists and ties in Example 2 in which unranked project centers / students are not listed and entities ranked in the same level are enclosed under the same brackets.

## 4.4 Introducing Cohort Construction Model

Our SPC optimization model considers cohort or team formation. We incorporates the notion of a cohort in the model via goal programming techniques. Cohort *attribute* that project center directors deem desirable in creating student teams can be gender or academic major, etc. Each attribute contains multiple values, called *level* of that attribute, for example, student major attributes contains a variety type of academic majors. We introduce a squared deviation penalty weighted by overassignment and underassignment weights to the objective function in formulation (4.1) to minimize the overassignment and underassignment of students deviated from desirable cohort targets. We present the quadratic model formulation of stable SPC with cohort construction in formulation (4.2) and notation related to the model in Table 4.3.

### Quadratic Model Formulation of Stable SPC with Cohorts

$$
\text{maximize} \quad \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} x_{sp} + \gamma \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} u_{sp}x_{sp} - \sum_{a\in\mathcal{A}}\sum_{i\in\mathcal{I}^a}\sum_{p\in\mathcal{P}} \overline{w}_{aip}\overline{y}^2_{aip} - \sum_{a\in\mathcal{A}}\sum_{i\in\mathcal{I}^a}\sum_{p\in\mathcal{P}} \underline{w}_{aip}\underline{y}^2_{aip} \tag{4.2a}
$$

$$
\begin{aligned}
\text{such that} \quad & \text{constraint sets (4.1b)–(4.1f)}, \\
& \sum_{s\in\mathcal{S}} b_{sai}x_{sp} - \overline{y}_{aip} + \underline{y}_{aip} = t_{aip} \quad \forall\ a\in\mathcal{A},\ \forall\ i\in\mathcal{I}^a,\ \forall\ p\in\mathcal{P}, && \text{(4.2b)} \\
& \overline{y}_{aip} \in \{0, 1, \ldots, c_p - t_{aip}\} \quad \forall\ a\in\mathcal{A},\ \forall\ i\in\mathcal{I}^a,\ \forall\ p\in\mathcal{P}, && \text{(4.2c)}
\end{aligned}
$$

| Sets | |
|---|---|
| $\mathcal{A}$ | Set of cohort attributes, indexed by $a$ |
| $\mathcal{I}^a$ | Sets of levels in cohort attribute $a \in \mathcal{A}$, indexed by $i$ |
| **Parameters** | |
| $t_{aip}$ | Desired cohort target (number of students) of level $i$ in attribute $a$ for project center $p$ |
| $b_{sai}$ | Indicator with value of 1 if student $s$ has level $i$ in attribute $a$; 0 otherwise |
| $\overline{w}_{aip}$ | Penalty weight for overassignment students with level $i$ in attribute $a$ to project center $p$ |
| $\underline{w}_{aip}$ | Penalty weight for underassignment students with level $i$ in attribute $a$ to project center $p$ |
| **Decision Variables** | |
| $\overline{y}_{aip}$ | Integer value of overassignment students with level $i$ in attribute $a$ to project center $p$ |
| $\underline{y}_{aip}$ | Integer value of underassignment students with level $i$ in attribute $a$ to project center $p$ |

Table 4.3: Notation for quadratic cohort construction models.

$$\underline{y}_{aip} \in \{0, 1, \ldots, t_{aip}\} \quad \forall\, a \in \mathcal{A},\ \forall\, i \in \mathcal{I}^a,\ \forall\, p \in \mathcal{P}. \tag{4.2d}$$

Objective function (4.2a) maximizes total placement and weighted utility less squared weighted penalty for placements that undershoot/overshoot targeted cohort features. The first two components in the objective function and constraint sets (4.1b)–(4.1f) are the same as in formulation (4.1), where stability is enforced by constraint set (4.1e). The number of overassignments and underassignments are determined in constraint set (4.2b). Lastly, variable domains on the (non-negative) number of overassignments and underassignments are presented in (4.2c) and (4.2d).

For each attribute $a \in \mathcal{A}$, level $i \in \mathcal{I}^a$, and project center $p \in \mathcal{P}$, either of $\overline{y}_{aip}$ and $\underline{y}_{aip}$ is positive; that is, the positive number of underassignment students $\underline{y}_{aip}$ implies the overassignment student $\overline{y}_{aip}$ is none, and vice versa. If there are $t_{aip}$ students with level $i$ in attribute $a$ assigned to project center $p$ already, the overassignment deviation from the desire target cannot exceed $c_p - t_{aip}$. On the other hand, if there is no student with level $i$ in attribute $a$ assigned to project center $p$, the underassignment deviation from the desire target in consideration is at most $t_{aip}$. Thus, we conclude that the upper bound of $\overline{y}_{aip}$ is $c_p - t_{aip}$ and the upper bound of $\underline{y}_{aip}$ is $t_{aip}$. we next present the results of a comprehensive set of computational experiments comparing the performance of (`SPC1`), (`SPC2`), (`VVM`), and (`BBT`) on real and synthetic datasets.

## 4.5 Computational Experiments

We now discuss experiments to compare and contrast the computational performance of different stability constraint sets. We conduct a variety of experiments on real-world and

synthetic datasets on formulation (4.1) with stability constraint set (SPC1) as constructed by Algorithm 1, stability constraint set (SPC2) as constructed by Algorithm 2, stability constraint set (VVM), and stability constraint set (BBT). Additionally, we compare the computational performance of the Quadratic Stable SPC formulation (4.2) for cohort construction on real-world datasets. Notably, for the three new representations we use (NWS) to eliminate blocking pairs associated with waste, except for the WPI datasets from the SPC matching at Worcester Polytechnic Institute (WPI). We show that the constraints may be dropped in WPI datasets as the objective prioritizes student utility.

### 4.5.1 Computational Setup and Datasets

All experiments were run using Gurobi Optimization 9.1 (2020) and Python, with a maximum of 64 GB memory, under Red Hat Enterprise Linux version 7.2. Each model instance in the real datasets was run with the following Gurobi parameters: TimeLimit of 24 hours, OptimalityTol dual feasibility tolerance of 1E-9, and MIPGap optimality tolerance of 0. Our limited testing on the comparison of using Big-M conditions and indicator constraints, detailed in Appendix B, results in a superior performance of the latter. As a result, we model Big-M conditions in all stability representations using Gurobi indicator constraints. As we conducted extensive synthetic data experiments, each model instance was run with TimeLimit of one hour and MIPGap optimality tolerance of 1E-4.

We study the performance of our approaches on three data sets: WPI, HRT, and synthetic. We have multiple years of WPI data in which we test our model development with all stability representations and the quadratic model formulation (4.2) of stable SPC with cohorts. In addition, we compare our development with a stable model formulation in [32] on a set of their HRT randomized instances. Finally, we consider a more general scenario by generating a host of synthetic data instances to further understand the computational performance of each stability representation on runtime and MIPGap. Detailed descriptions of the datasets are provided in Table 4.4.

| Dataset | Source | $\lvert\mathcal{S}\rvert$ | $\lvert\mathcal{P}\rvert$ | $\lvert\mathcal{R}_s\rvert$ | $\lvert\mathcal{R}_p\rvert$ | (NWS) | #Runs | #Replicates | TimeLimit/Inst. |
|---|---|---|---|---|---|---|---|---|---|
| WPI | SPC matching at WPI | 928; 927; 1126 | 46; 47; 57 | $\leq \lvert\mathcal{S}\rvert$ | 3 | ✗ | 3 | - | 24 hr |
| HRT | [32] | 759 | 53 | 5 | 5-6 | ✓ | 1 | 30 | 1 hr |
| Synthetic | Randomly Generated | Refer to Table 4.5 | | | | ✓ | 210 | 10 | 1 hr |

Table 4.4: Description of datasets used in our experiment.

### 4.5.2 Experimental Design

We evaluate the performance of (SPC1), (SPC2), (VVM), and (BBT) on three datasets. Investigations into the performance of our approaches on the WPI datasets are found in

Section 4.5.3. These three years of WPI data detail the preference lists of undergraduate students to be matched to worldwide project centers in the esteemed IQP program at WPI. We study how all variants of the stability representations perform across the three years for formulation (4.1), the stable matching with incomplete lists and ties and utility-weighted objectives. We then leverage these results together with formulation (4.2) to compare the effect of the cohort construction across the three WPI datasets and all variants of the stability representations.

In Section 4.5.4, we compare all variants of the stability representations on the HRT dataset from [32], the moderate size of 30 hospital-resident instances, called *master list*, denoted "RDM-ML-1-5". Each instance contains 759 doctors, 53 hospitals, 775 available seats, and the ranking of the doctors, made based on their grades, distributed in $[1, 5]$. While the preference of both doctors and hospitals in this dataset is incomplete, ties appear only in hospital preferences over doctors. The number of hospital preference tiers is five and the number of doctor preference tiers is either five or six. To compare our models with those of [32] using settings that are as similar as possible, we turn off their preprocessing with ties on the hospital side and convert their HRT instances to .lp files to solve using the Python API of Gurobi.

| Parameter | Symbol | Levels |
|---|---|---|
| Number of Project Centers | $\lvert\mathcal{P}\rvert$ | 10, 30, 50 |
| Project Center Capacity | $c_p$ | 10, 15, 20 |
| Percentage of Students | $\%\lvert\mathcal{S}\rvert$ | 50%, 75%, 100%, 125%, 150% |
| Number of Student Preference Tiers | $\lvert\mathcal{R}_p\rvert$ | $5, \ldots, \max_{\leq\lvert\mathcal{P}\rvert}\{15, 25\}$ |
| Popularity Among Project Centers | `popularity` | no, yes |

Table 4.5: The list of five parameters in the synthetic dataset.

In Section 4.5.5 we perform the full factorial design of parameters listed in Table 4.5 to study the effect of varying data dimensions on synthetic dataset and measure the performance of each variant of stability representations. Given a single dataset with $|\mathcal{P}|$ project centers, each with an equal capacity $c_p$, the percentage of student is $\%|\mathcal{S}|$, the number of students is $|\mathcal{S}| = \lfloor |\mathcal{P}| \times c_p \times \%|\mathcal{S}| \rfloor$. The number of student preference tiers is the number of ranks whereby students can place project centers, where placing project centers in the same tier indicates ties in their preference list. The `popularity` indicates whether the popularity among project centers is either *uniform* if the likelihood that any student select any project center into their preference list is equal across all project centers, or *nonuniform* if the likelihood that students select each project center into their preference list varies across all project centers due to differing popularity levels for each project center. We detail the generation of both popularity types in Appendix D.

Without loss of generality, we assume $r_{sp}$ represents the integer ordinal preference of student $s$ for project center $p$, where $r_{sp} \in \{1, 2, \dots\}$ and that smaller values of $r_{sp}$ indicate a more preferable project center for student $s$. We define the cardinal preference value for student utility as $q_{sp} := \frac{1}{r_{sp}}$ for the entirety of our computational experiments.

To study the effect of various stability representations for WPI datasets, we apply formulation (4.1) with the utility $u_{sp}$ set to the summation of weighted student utility and weighted project center utility, $\gamma_1 q_{sp} + \gamma_2 k_{sp}$. The values of $\gamma_1$ and $\gamma_2$ are carefully chosen so that the model preemptively optimizes in the following order: i) maximize student placement, ii) maximize student utility, and iii) maximize project center utility. We seek to determine coefficients $\gamma_1$ and $\gamma_2$ such that the strict ordering can be maintained. For example, we choose the values of $\gamma_1$ and $\gamma_2$ such that a) the contribution to the objective function of total project center utility ($3^{\text{rd}}$ component) is strictly less than the minimum contribution from placing any single student of the student utility ($2^{\text{nd}}$ component), b) the contribution to the objective function of total student utility ($2^{\text{nd}}$ component) is strictly less than the minimum contribution of a placement of any single student ($1^{\text{st}}$ component), and finally, c) the contribution to the objective function of total student utility and total project center utility together is strictly less than the minimum contribution of a placement of any single student. The last condition ensure that we obtain an objective value that reflects the number of student placements plus a small quantity of desired matching conditions.

We apply the same parameters $u_{sp}$, $\gamma_1$ and $\gamma_2$, using the stable solution results from the stable SPC formulation (4.1) of the WPI datasets as a warm start to study the performance of the quadratic cohort formulation (4.2a)–(4.2d). Our selection of penalty weights $\overline{w}_{aip}$ and $\underline{w}_{aip}$ retains the prior lexicographic ordering of the stable SPC formulation (4.1), and then preemtively minimize penalty of the squared deviation from desired cohort target in the last order. For our experiments, we set all $\overline{w}_{aip}$ to zero and set all $\underline{w}_{aip}$ to be equal to one another. We provide greater details on how we carefully derived coefficients used in the objective function of each dataset in Appendix E.

### 4.5.3 Experiments with WPI Datasets

Worcester Polytechnic Institute (WPI) is a private technological university in the Northeast United States that features a project-based and globally engaged curriculum. Perhaps the most distinctive program at WPI[2] is the Interactive Qualifying Project (IQP), which

[2]WPI global programs have been recognized by the Princeton Review as the most successful study-aboard program in the nation and also awarded the prestigious Bernard M. Gordon Prize from the United States National Academy of Engineering, which is the highest level of engineering honor there is in the United States, for Innovation in developing effective engineering leadership in 2016.

provides students an opportunity to apply their technical domain knowledge and skills to real-world projects. As part of the undergraduate degree requirement, WPI students have the opportunity to complete their IQP in Worcester or go to one of approximately 50 off-campus project centers, most of which are situated around the world. Those who choose an off-campus IQP go through a competitive selection process, followed by rigorous cultural preparation. Upon the arrival of students at each project center, students are formed into teams to work on a project. For many years the problem was solved manually, however, from 2018 onward, they used a version of our approach to recommend the matching of students to off-campus project centers while arriving at a final decision only after review by project center directors and careful examination by the program administrators.

The present mechanism requires students to rate each project center in three tiers that are assigned student utility weights of 1, 0.5, and 0, respectively. In an effort to increase access, students are required to choose at least three project centers in the first tier and at least six options in the first and second tiers combined. This process generated WPI datasets for the academic years of 2017–2018, 2018–2019, and 2019–2020. The complete result of stable SPC formulation (4.1) when varying stability representations across three years of the WPI datasets is shown in Table 6.1 of Appendix F.

**Comparing Performance of Stability Representations on WPI Datasets.** Table 4.6 reveals the result of stable SPC formulation (4.1) with varying stability representations across three years of the WPI datasets. By enforcing stability, all blocking pairs are eliminated. Across all three datasets, all four methods exhibit similar performance, as evidenced by the integrality gap which for all methods is below 2%, many methods found strong integer feasible solutions in times well earlier than the allotted run time, and the stable outcomes of all methods for the year 2018–2019 are solved to optimality. Thus, while we observe the following trends, we emphasize that all methods perform reasonably well. Some overall trends include that (`SPC2`) is consistently the sparsest representation of stability, both in the relative lack of nonzeros and density ratio[5]; this is perhaps related to the relative speed of building the model with Algorithm 2.

[5]model density is defined as $\frac{\#\text{Nonzeros}}{\#\text{Rows}\times\#\text{Columns}}$

| Year | 2017-2018 | | | | | 2018-2019 | | | | | 2019-2020 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metric | Stability Constraint Type | | | | | Stability Constraint Type | | | | | Stability Constraint Type | | | | |
| | None | SPC1 | SPC2 | VVM | BBT | None | SPC1 | SPC2 | VVM | BBT | None | SPC1 | SPC2 | VVM | BBT |
| Objective Value | **928.98** | **927.96** | 926.96 | 927.96 | 926.96 | **927.99** | **927.99** | **927.99** | **927.99** | **927.99** | 1,126.90 | 1,106.87 | **1,107.86** | 1,107.85 | 1,106.86 |
| Best Bound | **928.98** | **927.96** | 928.98 | 927.98 | 928.97 | **927.99** | **927.99** | **927.99** | **927.99** | **927.99** | 1,126.90 | 1,124.88 | 1,126.89 | **1,124.58** | 1,126.84 |
| MIPGap (%) | **0** | **0.0003** | 0.2172 | 0.0021 | 0.2167 | **0** | **0** | **0** | **0** | **0** | 0 | 1.6275 | 1.7173 | **1.5098** | 1.8051 |
| Best Incumbent Time | 0.2 | 75,264 | 28,319 | 5,763 | 43,313 | 0.11 | 46.26 | **8.25** | 97.41 | 78.17 | 0.14 | 86,400 | 85,305 | 31,707 | 86,400 |
| Run Time | 0.2 | 86,400 | 86,400 | 86,400 | 86,400 | 0.11 | 46.26 | **8.25** | 97.41 | 78.17 | 0.14 | 86,400 | 86,400 | 86,400 | 86,400 |
| Total Students | 928 | 928 | 928 | 928 | 928 | 927 | 927 | 927 | 927 | 927 | 1,126 | 1,126 | 1,126 | 1,126 | 1,126 |
| Tier-1 Placements | 885 | **853** | **863** | 849 | **863** | **927** | **927** | **927** | **927** | **927** | 1,049 | **988** | **976** | 951 | 965 |
| Tier-2 Placements | 43 | **74** | 63 | 78 | 63 | 0 | 0 | 0 | 0 | 0 | 77 | 118 | **131** | 156 | 141 |
| Unassignments | 0 | **1** | 2 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 20 | **19** | **19** | 20 |

Table 4.6: Comparison of SPC models with different stability constraint sets on three WPI datasets.

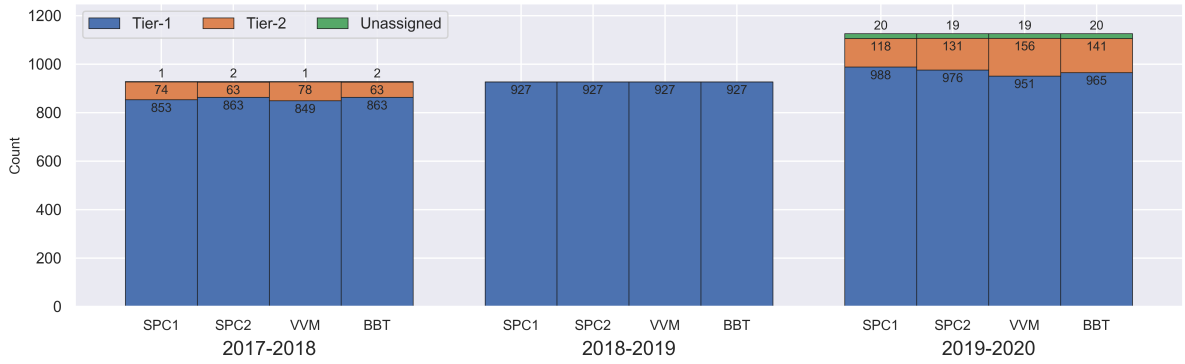



Figure 4.2: Placement comparison of SPC models with different stability constraint sets on three WPI datasets.

For the year 2018–2019, it is clear that while all methods perform reasonably well (perhaps due to the structure of the dataset), no approaches are superior to (`SPC2`) in terms of model build time, runtime, and shortest time to find the best incumbent. For the year 2017–2018, (`SPC1`) outperforms the other methods in terms of quality of solution, the smallest optimality gap, and total number of students placed. For the year 2019–2020, (`SPC2`) slightly outperforms the others in terms of solution quality, while being slightly outperformed by (`VVM`) in terms of optimality gap.

In summary, all stability constraint sets perform comparatively well on the three WPI datasets as they result in comparatively the same matching quality in Figure 4.2. While the results were close, (`SPC2`) results in the best objective values in two out of three years, 2018–2019 and 2019–2020. Outside of these trends, a case for best overall performing constraint set on these experiments can be made for (`SPC2`), (`SPC1`), and (`VVM`), but not (`BBT`).

**Computational Results of Quadratic Cohort Models.** We now study the construction of cohorts and corresponding computational performance of using the SPC quadratic goal programming formulation with stability constraints. We construct cohorts where all

project center directors request at least 20% or more of the students are female and at least 20% or more are computer science (CS) majors. Thus, objective function (4.2a) is penalized via underassignment only when the total number of underassigned female students or underassigned CS students is below $\lceil 0.2 \times c_p \rceil$. The complete result of quadratic stable SPC formulation (4.4) with cohorts when varying stability representations across three years of the WPI datasets is shown in Table 6.2 Appendix F.

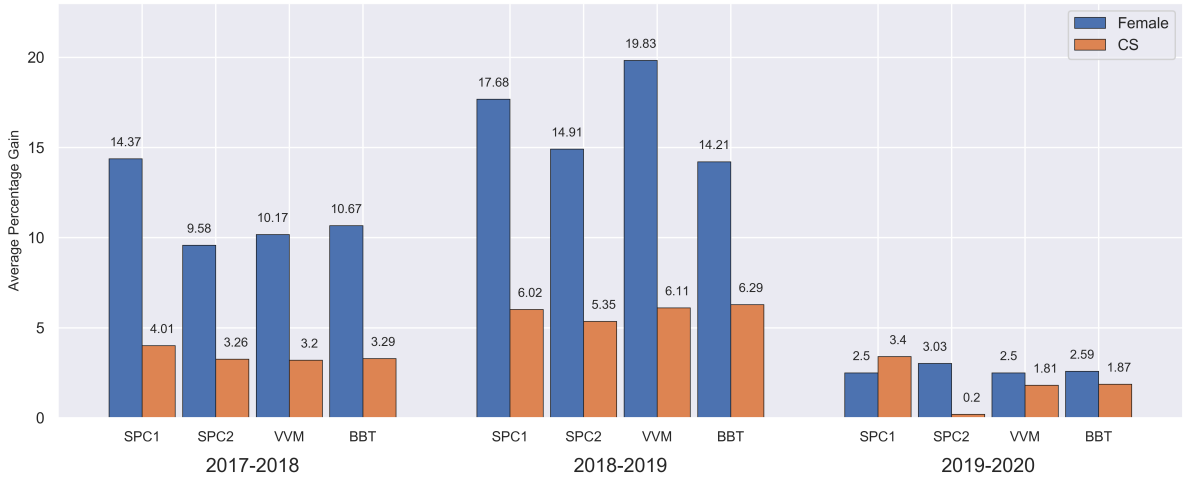



Figure 4.3: Average gain in percentage of underassignment deviation across project centers with underassigned female or CS students by the quadratic SPC formulations with cohorts applying different stability constraint set on three WPI datasets.

The formulation (4.2) can create cohorts that, while maintaining the same number of total students placed, have fewer underassignments of females and CS students, respectively. Figure 4.3 shows positive average percentage improvement in the number of underassignment female and CS students across all variants of stability representations and all years. The average loss in term of the objective function value compared with the corresponding objective function value solution without cohorts for all methods in all the three years is less than 0.02%. The results suggest that the formulation is not only computationally tractable for the purposes of SPC with cohorts, but that it is reasonably effective in constructing more diverse cohorts.

### 4.5.4 Comparing Performance of Stability Representations on the HRT Dataset

We implement the model from Section 3.2 of [32] in our experiments, which we call (`HRT1`). Because a main difference between our two works is that we consider weighted utility, we adapt the objective function in (6) of [32] to include weighted project center utility.

In light of this, we consider the following which is an adjustment of our objective function (4.1a), where the utility $u_{sp}$ is set to the project center utility $k_{sp}$, and the weight $\gamma$ is chosen so that it first prioritizes maximizing student placement, followed by maximizing project center utility:

$$\text{maximize} \quad \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} x_{sp} + \gamma \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} k_{sp}x_{sp}. \tag{4.3}$$

Here, the weight $\gamma$ is defined as $\frac{1}{|\mathcal{S}|+\varepsilon}$, where we set the value of $\varepsilon$ to be 1e$-$6, and the utilities $k_{sp}$ are defined as the reciprocal of the tier for student $s$ for project center $p$. The stability constraint set in [32] is (`HRT`) which, as stated in Section 4.3, is equivalent to (`BBT`).

In discussing the results of Table 4.7, we aggregate the results of the 30 selected master-list instances by averaging over the runtime per stability representation. We summarize the results in Table 4.7; (`SPC1`) results in the best average runtime, followed by (`VVM`), (`SPC2`), (`HRT1`), and (`BBT`), respectively. Out of the 30 instances, 17 of (`SPC2`), 5 of (`SPC1`), 5 of (`VVM`), and 3 of (`HRT1`), respectively, result in the best runtime. The performance of (`BBT`) and (`HRT1`) are comparatively the same. This provides at least limited evidence that our introduced stability constraints may perform well in the real-world HRT instances that maximize stable matching with utility.

| Method | SPC1 | SPC2 | VVM | BBT | HRT1 |
|---|---|---|---|---|---|
| Average Runtime (s) | **628.65** | 726.09 | 685.28 | 1,600.62 | 1,563.36 |
| #Best Runtime (out of 30 instances) | 5 | **17** | 5 | - | 3 |

Table 4.7: Performance comparison of HRT1 models with different stability constraint sets on the HRT dataset.

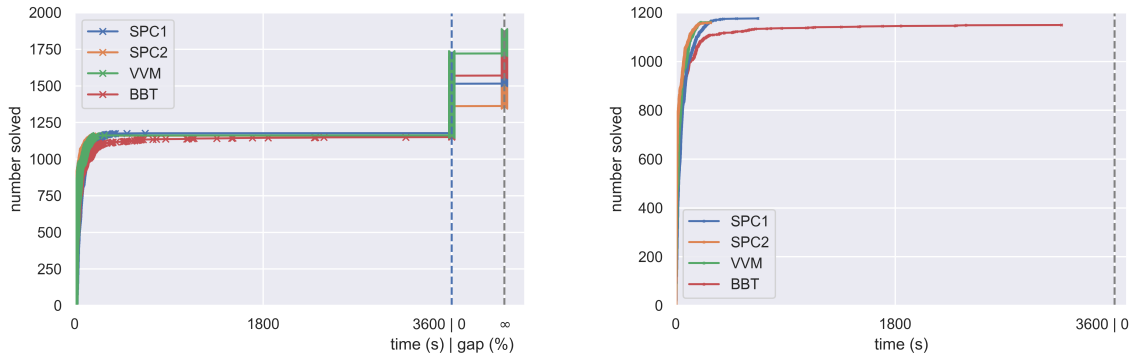
Stability representations (`SPC1`), (`SPC2`), and (`VVM`) have exhibited competitive performance on the WPI and HRT datasets. We now pursue an in-depth investigation of their performance on the synthetic dataset, to both confirm our understanding and investigate further properties of the data.

### 4.5.5 Comparing Performance of Stability Representations on Synthetic Datasets

We apply formulation (4.1) with the utility $u_{sp}$ set to the weighted student utility in which the value of $\gamma$ is chosen so that the model first prioritizes maximizing student placement, followed by maximizing student utility. The full factorial experiment over all levels yields 210 runs. We create 10 randomized data instances, or replicates, for each unique set of parameters to reduce variation and account for error. Of these 210 runs, we remove 23

indeterminate runs in which the absolute runtimes of all pairs of the 10 instances across all stability representations are less than 0.5 seconds. As a result, we focus our study on the remaining 187 runs (1,870 instances in total). We examine both the performance of the four stability representations on the 1,870 instances and on the 187 runs. Both results consistently suggest the circumstances under which each representation excels.

The solution status for each of the 1,870 instances is either solved to optimality, suboptimal (timed out), or no solution found within the time limit; each type of instance is depicted in the first, second, and third segment, respectively, according to vertical dotted gray line in Figure 4.4a and the corresponding Table 4.8. The performance of the four methods is reported by the runtime if the solution is solved to optimality, by the MIPGap otherwise. We also report average runtime/MIPGap (AvgTime/AvgMIPGap) and maximum runtime/MIPGap (MaxTime/MaxMIPGap) for each method where applicable in Table 4.8.




(a) (b)

Figure 4.4: (left) Cumulative distribution plot comparing each stability constraint set on 1,870 instances. (right) Close-up of first segment of the cumulative distribution plot.

| Method | Runtime ≤ 3600 s | | | Timed out | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Optimal (MIPGap < 1E-4) | | | Suboptimal (MIPGap ≤ 100%) | | | | No Solution Found | |
| | AvgTime | MaxTime | Height | AvgMIPGap | MaxMIPGap | Increment | Height | Increment | Height |
| SPC1 | 54.5 | 672.9 | **1,176** | 0.02% | 0.06% | 338 | 1,514 | 356 | 1,870 |
| SPC2 | **26.0** | 285.5 | 1,159 | 0.04% | 0.17% | 202 | 1,361 | 509 | 1,870 |
| VVM | 35.2 | **282.4** | 1,161 | **0.01%** | **0.03%** | **559** | **1,720** | **150** | 1,870 |
| BBT | 71.7 | 3163.1 | 1,149 | **0.01%** | **0.03%** | 420 | 1,569 | 301 | 1,870 |

Table 4.8: Comparison of the four stability methods with respect to the cumulative distribution of performance. *Height* is the height as measured against the $y$-axis of the cumulative plot of a given method up to the specified segment.

Based on Table 4.8, the first segment shows that all methods, except (BBT), have roughly similar performance with smaller average runtime and maximum runtime of all optimal instances. It can also be seen that (SPC2) is the superior method because its

instances result in the smallest average runtime and are solved to optimality within 300 seconds, while the instances of (VVM) also perform well solved under 300 seconds with the smallest maximum runtime and (SPC1) solves the greatest number of instances. The second segments show that, for instances that are not solved to optimality, all methods exhibit roughly similar performance with average MIPGap values of just above zero. This implies that the suboptimal instances of all methods have virtually zero percent gap, but are not technically solved optimality due to secondarily favoring some combination of utility. The timed-out instances of (VVM) find the greatest number of feasible solutions in this section. On the other hand, while (SPC2) outperforms others when the instances can be solved to optimality, it finds the fewest feasible solutions in this section. Finally, the third segment shows that (SPC2) is the method that has the most instances where no feasible solutions were found within the time limit. We observe instances that timed out for (SPC2) are primarily those that have a relatively large value of $\%|\mathcal{S}| > 100\%$, whereas (VVM) is the method most likely to find a feasible solution, followed by (BBT), and (SPC1), respectively.

We now examine the performance of the four stability representations on the 187 runs. The number of runs where at least one instance out of 10 instances has a feasible solution (not necessarily solved to global optimality) of each method over all 187 runs is as follows: (SPC1): 169, (SPC2): 160, (VVM): 185, and (BBT): 177.

We categorize each of the 187 runs into three classes: (1) all 10 instances solved to global optimality, (2) some solved to global optimality and some timed out, and (3) all 10 instances timed out. Runtime is considered in the first two classes when some instances solved to global optimality and MIPGap is considered when all instances timed out (MIPGap $>$ 1E$-$4). Note that we omit instances where the method cannot find a feasible solution within the time limit from the ensuing discussion. Table 4.9 summarizes the best method with the lowest average runtime/MIPGap resulted from the 187 runs; (SPC2) performs the best when all instances are solved to global optimality, whereas (VVM) tends to perform best when some instances time out.

We now consider the effect of each parameter on the performance of each stability constraint set for the purpose of making recommendations for the best methods across various parameter settings and levels. Our initial exploratory data analysis shows that the most important parameters on the runtime are in the following order: $\%|\mathcal{S}|, |\mathcal{R}_p|, c_p, |\mathcal{P}|$, and popularity. All instances in Class 1 feature $\%|\mathcal{S}| \leq 100$. Class 2 includes instances with $\%|\mathcal{S}| \geq 100$ and a combination of small to large levels of $|\mathcal{P}|$ and $c_p$. On the other hand, Class 3 features instances with $\%|\mathcal{S}| > 100$ and medium to large levels of $|\mathcal{P}|$ and $c_p$. We analyze the data and identify general trends according to certain parameters.

The (SPC2) method clearly outperforms others in Class 1. Classes 2 and 3 feature

| | Metric | SPC1 | SPC2 | VVM | BBT | Total |
|---|---|---|---|---|---|---|
| Class 1: all 10 instances solved to optimality | Runtime | 1 | 89 | 13 | 12 | 115 |
| Class 2: some solved to optimality and some timed out | Runtime | 0 | 1 | 42 | 4 | 47 |
| Class 3: all 10 instances timed out | MIPGap | 0 | 2 | 12 | 11 | 25 |
| | | 1 | 92 | 67 | 27 | 187 |

Table 4.9: Best method with the lowest average runtime result from all 187 runs.

larger levels of parameters that are more difficult to solve with $\%|\mathcal{S}| > 100$. The (`VVM`) method clearly outperforms other in Class 2, while (`VVM`) and (`BBT`) perform relatively better in Class 3. As the problem size tends to grow larger from Class 1 to Class 3, so does the computational complexity. The analysis according to our setting shows the range of parameter values whereby each method performs well in Table 4.10.

| Method | $\lvert\mathcal{P}\rvert$ | & | $c_p$ | & | $\%\lvert\mathcal{S}\rvert$ | & | $\lvert\mathcal{R}_p\rvert$ |
|---|---|---|---|---|---|---|---|
| `SPC1` | $\leq 40$ | | $\leq 15$ | | $\leq 125$ | | $\leq 10$ |
| `SPC2` | – | | – | | $\leq 100$ | | – |
| `VVM` | – | | $\leq 15$ | | $\leq 150$ | | $\leq 20$ |
| `BBT` | $\leq 40$ | | $\leq 15$ | | $\leq 125$ | | – |

Table 4.10: Summary of parameters where respective methods tend to outperform others.

In summary, across all metrics, (`SPC2`) clearly outperforms the others when there are enough seats for all students, that is, $|\mathcal{S}|$ does not exceed total capacity. However, the performance of (`SPC2`) can degrade fairly quickly when the number of students begins to exceed the capacity. All other methods have less of a clear demarcation. For particularly difficult instances, such as when $\%|S|$ is relatively high and large values of either $|\mathcal{P}|$ or $c_p$ or both, it remains unclear under which parameters different methods outperform others. In this case, our result suggests that (`VVM`) is the most well-rounded method, while (`BBT`) and (`SPC1`) also perform well when all parameters are in moderate ranges.

## 4.6 Conclusions and Future Work

We study many-to-one stable matching with ties and incomplete lists via integer optimization. We use lexicographic optimization to construct an objective function that allows the model to optimize each component of the objective function in the desired order, for example, total placement, utility, and cohorts. We introduce several new representations of stability and conduct a comprehensive study on their computational performance on real and synthetic datasets. Additionally, we incorporate the concept of cohorts via a goal programming technique that penalize deviations in a quadratic manner to steer the search toward more diverse cohorts.

Computational experiments on real-world datasets demonstrate that our new stability representations generate a fairly good matching, and the quadratic deviation model is computationally tractable for finding a good stable match outcomes with cohort construction. Specifically, (`SPC1`), (`SPC2`), and (`VVM`) are promising in SPC matching and their performance outperforms existing methods when utility is considered in the objective function. Furthermore, limited simulation experiments demonstrates that (`SPC2`) is a promising method for typical real-world applications where the number of applicants does not exceed the number of seats, which can be seen in school choice problems and hospital-residency matching. In all cases, stability constraint set (`VVM`) is an all-around method for stable matching applications.

Future work includes testing our approaches on additional datasets, as well as studying the effect of different models or stability constraint sets on a variety of objective functions or utility functions in the objective function could told more insights. Researching the transformation of our binary variables into integer variables presented in Algorithms 1 and 2 in a similar manner to the advanced models in [18], as well as extending the advanced models to accept the settings of utility functions in the objective function may increase the computational performance. In Chapter 5, we present a linearization of the quadratic stable SPC formulation (4.2) with cohorts and its generalization. Our limited testing shows the superior performance of the linear stable SPC with cohort formulation over the quadratic version. Finally, we suggest further investigation on our disaggregate constraints (`PW`) to determine how they could be further leveraged to obtain computational efficiencies.

# Chapter 5

# A Reformulation Technique to Solve Polynomial Optimization Problems with Separable Objective Functions of Bounded Integer Variables

## Introduction

Integer optimization (IO) has seen widespread use in solving challenging decision problems due to its expressivity and ability to characterize constrained decisions under an objective function to be optimized. Advances in algorithmic development and computing over the past several decades have seen profound increases in the potential of commercial optimization solvers [1]. Even so, many real world decision problems with nonconvex functions represent vexing challenges for global optimization solvers.

We study a broad class of polynomial integer nonlinear optimization (PINLO) problems:

$$\begin{array}{lll} \text{maximize} & \sum_{k=1}^{p}\sum_{j=1}^{n_x} c_{kj}x_j^k + \sum_{\ell=1}^{n_y} c_\ell y_\ell & \\ \text{such that} & \sum_{j=1}^{n_x} a_{ij}x_j + \sum_{\ell=1}^{n_y} a_{i\ell}y_\ell \le b_i, & i = 1,\ldots,m, \\ & x_j^L \le x_j \le x_j^U,\ x_j \in \mathbb{Z}_{\ge}, & j = 1,\ldots,n_x, \\ & y_\ell^L \le y_\ell \le y_\ell^U,\ y_\ell \in \mathbb{R}, & \ell = 1,\ldots,n_y, \end{array} \tag{5.1}$$

where the objective function is a separable polynomial function, constraints are linear $\forall\ i = 1,\ldots,m$, $x_j^L$ and $x_j^U$ are lower and upper bounds of nonnegative integer $x_j\ \forall\ j = 1,\ldots,n_x$, and $x_j^k$ is the $k^{\text{th}}$ degree polynomial of $x_j\ \forall\ j = 1,\ldots,n_x,\ \forall\ k = 1,\ldots,p$. Continuous variables $y_\ell$ may appear in linear expressions with lower and upper bounds of $y_\ell^L$ and $y_\ell^U\ \forall\ \ell = 1,\ldots,n_y$, respectively. We will refer to this specific class of PINLO

formulation (5.1) as PINLO through this study.

Many real-world applications of nonlinear integer optimization features a quadratic objective function subject to a set linear constraints, often called mixed-integer quadratic programming (MIQP), and when there are no continuous variables, (pure) integer quadratic programming (IQP) [39, 40]. These problems can be viewed as a generalization of (mixed-)integer linear optimization with a nonlinear objective function [41]. It thus encompasses all integer linear optimization (ILO) problems, including applications in scheduling, planning and network flows. PINLO appears (often in quadratic form) in a variety of applications, such as quadratic knapsack problems [42–45], two-stage quadratic stochastic programs [46], multicommodity network flow problems [47], portfolio selection [48–51], heat transfer process [52], and goal programming with quadratic deviation penalties [24, 53].

Many exact methods for solving formulation (5.1) have been proposed. Branching-based methods include branch-and-bound (BB) [3, 54–56], branch-and-reduce [57–61], and $\alpha$BB [62–65]. Methods that reformulate PINLO problems into ILO problems include the use of sets of binary variables with a certain special structure, called special ordered sets of type 2 (SOS2) [66, 67], as well as piecewise linear functions [68–73].

Other exact algorithms with linear constraints and a nonlinear objective function include the use of concavity cutting planes [74–76], a hybrid method that combines dynamic programming and branch-and-bound approaches to produce an algorithm for solving separable discrete optimization [77], an algorithm to simplify nonseparable functions [78], a Lagrangian decomposition technique [79], and expressing nonconvexity in the objective and constraint functions as the sum of nonconvex univariate functions [80]. Moreover, [41, 81–84] review multiple global optimization approaches for the general nonconvex optimization problem.

This study presents two reformulations of a class of polynomial integer nonlinear optimization model (PINLO) with separable and bounded integers and any degree of polynomial to integer linear optimization (ILO). We derive an algebraic identity presented in Section 5.1.1 to use for the first linearization approach which expresses values of polynomial integers as the summation of cumulative weights. The second linearization approach usees precomputed weights to express polynomial values. We demonstrate the equivalence of PINLO formulation (5.1) and the proposed ILO reformulations. We subsequently conduct the comparative computational experiments of PINLO and ILO reformulations on a synthetic dataset and a real-world goal programming application. The computational experiments reveal that the ILO reformulations outperform PINLO using the state-of-the-art commercial solvers, BARON [61, 85] and Gurobi [38]. Additionally, the cumulative-weight ILO outperforms precomputed-weight ILO in larger problem dimensions.

The remainder of this study is structured as follows. Section 5.1 presents theoretical

concepts of the linearization of the specific class of PINLO formulation (5.1). Section 5.2 presents computational experiments of quadratic penalty goal programming for matching application and simulations to compare our approaches with leading global optimization solvers. Lastly, Section 5.3 concludes our study.

## 5.1 Linearization of PINLO

We introduce a linearization of PINLO formulation (5.1) using recurrence relations and the series identity of positive integer powers of degree $p$. We first present the identity for our linearization in Section 5.1.1 which can be derived from Faulhaber's formula. We next outline the steps for PINLO-to-ILO linearization which expresses a polynomial of integer variables as the cumulative summation of the derived formulations in Section 5.1.2. Lastly, we present another linearization of PINLO-to-ILO which expresses the polynomial of integer variables as the summation of precomputed weights multiplied by binary variables in Section 5.1.3.

### 5.1.1 Derivation of The Finite Summation Identity

To find an expression of the $p^{\text{th}}$ power of any positive integer $n$, we explore the reformulation of polynomial terms using the finite summation identity that equals $n^p$. One way to formulate $n^p$ as a finite summation of $n$ summation terms is via the recursion of the finite summation of the $p^{\text{th}}$ power of the first $n$ positive integers. The finite summation identity of $n^2$ is equal to the sum of the first $n$ positive odd integers,

$$n^2 = \sum_{i=1}^{n} (2i - 1),$$

which can be derived from this well-known identity related to the binomial theorem,

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}.$$

In general, we can use *Faulhaber's formula* [86] to express the sum of the $p^{\text{th}}$ power of the first $n$ positive integers in terms of the Bernoulli numbers, as follows:

$$\sum_{i=1}^{n} i^p = \frac{1}{p+1} \sum_{k=0}^{p} (-1)^k \binom{p+1}{k} B_k n^{p+1-k}, \tag{5.2}$$

where $B_k$ is the Bernoulli number with $B_1 = -\frac{1}{2}$.

Another way, perhaps simpler, is to apply the binomial theorem to derive the finite

summation of $n^p$. For any integer $i$, the binomial theorem states that

$$(i+1)^{p+1} = \sum_{k=0}^{p+1} \binom{p+1}{k} i^k. \tag{5.3}$$

We next obtain the following expression:

$$(i+1)^{p+1} - i^{p+1} = \sum_{k=0}^{p} \binom{p+1}{k} i^k. \tag{5.4}$$

Substituting $i = -1, -2, \ldots, -n$ to (5.4) and summing, the left-hand side telescopes to $-(-n)^{p+1}$ and we obtain

$$-(-n)^{p+1} = \sum_{k=0}^{p} (-1)^k \binom{p+1}{k} \left[1^k + \cdots + n^k\right]. \tag{5.5}$$

Multiplying both sides of (5.5) by $(-1)^p$, we obtain the following identity

$$n^{p+1} = \sum_{k=0}^{p} (-1)^{p+k} \binom{p+1}{k} \sum_{i=1}^{n} i^k, \tag{5.6}$$

which can be used to express the $p^{\text{th}}$ power of any positive integer $n$ in term of the summation of $n$ terms as

$$n^p = \sum_{k=0}^{p-1} (-1)^{p-1+k} \binom{p}{k} \sum_{i=1}^{n} i^k, \tag{5.7}$$

For example, $n^2 = (-1)(1^0 + 2^0 + \cdots + n^0) + (2)(1^1 + 2^1 + \cdots + n^1) = \sum_{k=1}^{n}(2k-1)$. We next apply (5.7) to linearize any positive integer power of degree $p$.

### 5.1.2 PINLO-to-ILO Linearization via Cumulative Summation to Express Powers of Integer Variables

We introduce a linearization of polynomial nonlinear terms of bounded integers. We outline the reformulation of the PINLO-to-ILO reformulation in three steps. The first step is to define binary variables that collectively represent all integer variable values, and through constraints enforce the integer variable values by summing over binary variables. The second step is to ensure that, depending on the value of the integer variable, the appropriate binary variables are activated. The last step is the derivation of appropriate weights that, when aggregated, are an equivalent representation of the variable values in the respective polynomial expression.

**First step of reformulation.** We express the value of each nonnegative integer $x_j$ $\forall\ j = 1, \ldots, n_x$ in formulation (5.1) as the summation of binary variables $x_{j,d}$ where the index $d \in [1, x_j^U] \subset \mathbb{Z}_{\geq 0}$ counts the occurrence of the binary variables for $x_j$. We define

binary variables $x_{j,d}$ taking a value of 1 if $x_j \geq d$ for integer values $d \geq 1$, respectively, and 0 otherwise. This means that if $x_j = k$ for some positive integer $k$, then all $k$ binary variables $x_j^1, \ldots, x_j^k$ are activated to 1 because $x_j \geq k \geq \cdots \geq 1$. Fig. 5.1 demonstrates the activation of $x_{j,d}$ when $x_j = 8$. The activation is applied through constraint sets (5.9) and (5.10) that we define next.

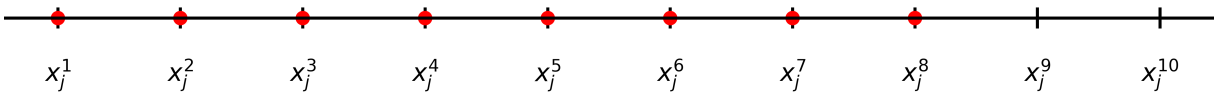



Figure 5.1: Illustrating activated $x_{j,d}$ in red when $x_j = 8$.

**Second step of reformulation.** The integer decision variables $x_j$ are now represented as the sum of $x_{j,d}$, where

$$x_j = \sum_{d=1}^{x_j^U} x_{j,d}. \tag{5.8}$$

We use disjunctive constraints to ensure that the binary variables take appropriate values to represent the integer variable values $x_j$. That is, through the following two sets of inequalities, $x_{j,d} = 1$ if and only if $x_j \geq d$ and 0 otherwise, for integer value $1 \leq d \leq x_j^U$:

$$x_j - d \leq M_{j,d} x_{j,d} - 1 \quad \forall\, j = 1, \ldots, n_x,\ \forall\, d = 1, \ldots, x_j^U, \tag{5.9}$$

$$x_j - d \geq m_{j,d}(1 - x_{j,d}) \quad \forall\, j = 1, \ldots, n_x,\ \forall\, d = 1, \ldots, x_j^U. \tag{5.10}$$

The value of each $M_{j,d}$ is the upper bound of $x_j - d + 1$ and the value of each $m_{j,d}$ is the lower bound of $x_j - d$. If $x_j \geq d$, $x_{j,d}$ must be one, otherwise constraint set (5.9) is violated. If $x_j < d$, $x_{j,d}$ must be zero, otherwise constraint set (5.10) is violated.

We note that if the lower bound of $x_j$ is strictly greater than zero, that is, $0 < x_j^L \leq x_j$, then it follows that $x_{j,d} = 1$ for $1 \leq d \leq x_j^L$ and, thus, we can drop the disjunctive constraints for binary variables $x_{j,d}$ for $d \leq x_j^L$.

**Final step of reformulation.** To enforce the polynomial value, we introduce special weights via the identity (5.7) that states the equivalence of the $p^{\text{th}}$ power of a positive integer number $n^p$ and the sum of $n$ positive integer numbers. Denote $w_{j,d}$ to be a weight value of $x_{j,d}$ $\forall\, d \in \{1, \ldots, x_j^U\}$ that takes a value of

$$w_{j,d} = \sum_{k=0}^{p-1} (-1)^{p+k-1} \binom{p}{k} d^k \quad \forall\, d \in \{1, \ldots, x_j^U\}, \tag{5.11}$$

so that their cumulative sum of the product of $w_{j,d}$ and $x_{j,d}$ over $d = 1, \ldots, x_j^U$ equals to $x_j^p$. Equivalently, we can view the weight $w_{j,d}$ as an increment of an additional unit increase of $x_j^p$.

Through the use of binary variables, disjunctive constraints, and weights according to (5.7), our reformulation (ILO1) transfers the polynomial expression $x_j^p$ to an equivalent linear expression:

$$x_j^p = \sum_{d=1}^{x_j^U} w_{j,d} x_{j,d} \quad \forall\ j = 1, \ldots, n_x. \tag{ILO1}$$

Our (ILO1) reformulation offers a new exact solution technique for solving PINLO problems which can be implemented directly using any state-of-the-art ILO solver. We demonstrate the equivalence of PINLO objective function and reformulated ILO objective function in Proposition 1.

**Proposition 1** *For $x_j \in \mathbf{X}_j$, $x_j^p = \sum_{d=1}^{x_j^U} w_{j,d} x_{j,d}$.*

**Proof.** Without loss of generality, assume $x_j = v$ for an arbitrary $v \in \{1, \ldots, x_j^U\}$:

$$\begin{aligned}
\sum_{d=1}^{x_j^U} w_{j,d} x_{j,d} &= \sum_{d=1}^{x_j^U} \left( \sum_{k=0}^{p-1} (-1)^{p+k-1} \binom{p}{k} d^k \right) x_{j,d} \\
&= \sum_{k=0}^{p-1} (-1)^{p+k-1} \binom{p}{k} \left[ 1^k \cdot x_{j,1} + \cdots + (x_j^U)^k \cdot x_{j,x_j^U} \right] \\
&= \sum_{k=0}^{p-1} (-1)^{p+k-1} \binom{p}{k} \left[ x_{j,1} + \cdots + v^k \cdot x_{j,v} \right], \ \{\text{by (5.9) and (5.10)}\} \\
&= \sum_{k=0}^{p-1} (-1)^{p+k-1} \binom{p}{k} \left[ 1 + \cdots + v^k \right] \\
&= v^p, \ \{\text{by (5.7)}\}
\end{aligned}$$

as the choice of $v$ was arbitrary in the domain of $x_j \in \mathbf{X}_j$, this completes the proof. ∎

If $x_j^L > 0$, then $x_{j,d} = 1$ for $d \leq x_j^L$. Therefore, the value of $x_j^p$ is at least $(x_j^L)^p$, that is

$$x_j^p \leq \sum_{k=0}^{p-1} (-1)^{p+k-1} \binom{p}{k} \left[ 1 + \cdots + (x_j^L)^k \right] = (x_j^L)^p.$$

Proposition 1 implies the equivalence of the objective functions of PINLO formulation (5.1) and ILO formulation (ILO1), that is,

$$\sum_{j=1}^{n_x} c_{kj} x_j^p = \sum_{j=1}^{n_x} c_{kj} \sum_{d=1}^{x_j^U} w_{j,d} x_{j,d}.$$

### 5.1.3 PINLO-to-ILO Linearization via Precomputed Weights to Express Powers of Integer Variables

We compare (ILO1) with an alternative reformulation that linearizes polynomial terms using precomputed weights for the value of polynomial integers. We express the $p^{\text{th}}$ power of $x_j$ in formulation (5.1) in terms of the summation of precomputed weights multiplied by binary variable $x_{j,q}$ $\forall\ q = 0, \ldots, x_j^U$ where $x_{j,q} = 1$ if $x_j = q$, and 0 otherwise. As a result, we can express $x_j^p$ as

$$x_j^p = \sum_{q=0}^{x_j^U} q^p x_{j,q}, \quad \forall\ j = 1, \ldots, n_x. \tag{ILO2}$$

To enforce (ILO2), we first ensure that exactly one binary variable is activated by adding the following constraint

$$\sum_{q=0}^{x_j^U} x_{j,q} = 1 \quad \forall\ j = 1, \ldots, n_x. \tag{5.12}$$

Then, we impose the relationship between $x_j$ and the binary variables in which $x_j = q$ $\forall\ j = 1, \ldots, n_x$ if $x_{j,q} = 1$ by adding the following constraint

$$x_j = \sum_{q=0}^{x_j^U} q x_{j,q} \quad \forall\ j = 1, \ldots, n_x. \tag{5.13}$$

We next present computational experiments that compare the performance of our ILO reformulation in Section 5.1.2 with PINLO formualtion (5.1) and the simple ILO reformulation in Section 5.1.3. We conduct experiments on two datasets: 1) a synthetic dataset, and 2) a real-world application using goal programming with quadratic penalty deviation to create cohorts (teams / groups) in a many-to-one stable matching of students to project centers.

## 5.2 Computational Experiments

### 5.2.1 Experimental Setup

We evaluate the performance of our ILO reformulation (ILO1) by comparing with PINLO formulation (5.1) and ILO reformulation (ILO2) using precomputed weights. We conduct the experiments on a synthetic dataset and a real dataset of a stable many-to-one matching application, presented in Chapter 4.4, that assigns students to project centers and creates cohorts, that is, groups of students having one or more particular features in common. All synthetic experiments were run via NEOS server [87–89] using BARON 21.1.13 [61, 85] to

solve PINLO and ILO. We also solve ILO using Gurobi Optimization 9.1 [38] and Python API with up to 64 GB memory, under Red Hat Enterprise Linux Server 7.3 with kernel version 3.10.0-514.x86_64. Each synthetic instance was run with time limit of 3 hours, MIP optimality gap tolerance of 0, Absolute MIP optimality gap of 0, and thread count to 1; the experiments on the matching application were run with time limit of 24 hours and MIP optimality gap tolerance of 0.

**Synthetic Dataset**

We perform a full factorial design by varying the parameters listed in Table 5.1. The domain of the integer variables is $[0, x^U]$, where the upper bound $x^U \in \{10, 100\}$. The degree of polynomial objective functions of integer variables in all instances is chosen to be a reasonable degree in polynomial problems $p \in \{2, 3, 5\}$, though it is not prohibitive to increase $p$ to much larger values.

| Parameter | Symbol | Levels |
|---|---|---|
| Integer variable upper bound | $x^U$ | $10, 100$ |
| Polynomial degree | $p$ | $2, 3, 5$ |
| Ratio of integer-to-total number of variables | $\frac{n_x}{n}$ | $0.5, 1$ |
| Density of objective coefficients | $\Delta$ | $0.5, 1$ |
| Number of constraints | $m$ | $25, 50, 75, 100, 150,$ $250, 500, 750, 1{,}000$ |
| Scale parameter for problem size | $\alpha$ | $0.5, 2$ |

Table 5.1: Parameters used for generating synthetic instances.

We select the values of model parameters in a similar manner as described in [90]. We set the density of constraints to be 50%. Coefficients in the left-hand side of the constraints are drawn randomly from a discrete uniform distribution $[1, 30]$ and constants in the right-hand side of the constraints are drawn randomly from a discrete uniform distribution $[30, 30 + \sum_{j=1}^{n} a_{ij}]$ for $i = 1, \ldots, m$. We set the number of constraints $m \in \{25, 50, 75, 100, 150, 250, 500, 750, 1{,}000\}$. The total number of variables $n$ is controlled by the scale parameter for the problem size $\alpha \in [0.5, 2]$ in which $n = \alpha m$.

We generate two classes of instances controlled by the ratio of the number of integer variables to the total number of variables. Instances are pure integer where all variables are integers when $\frac{n_x}{n} = 1$ and are mixed integer where the number of integer variables $(n_x)$ is half of the total number of variables when $\frac{n_x}{n} = 0.5$. More specifically, variables in mixed-integer instances consist of 50% integer variables, 25% continuous variables with values that are drawn randomly from a uniform distribution $[0, 10]$, and 25% binary

variables.

An objective function of all instances contains integer variables, each with degree up to $p$, as well as continuous and binary variables if the instance is a mixed-integer problem. For example, an objective function of pure integer instances contains up to $n \times p$ integer terms as each integer variables may have degree up to $p$, whereas an objective function of mixed-integer instances contains $n \times p$ integer terms and $\frac{1}{2}n$ continuous and binary terms. The density of nonzero elements in the objective function is controlled by a Bernoulli probability $\Delta \in \{0.5, 1\}$, where the values of the coefficients are drawn from a discrete uniform distribution $[0, 100]$. The combination results in 432 runs in total, for which we create three replicates and average over each run. We report the computational results of the synthetic dataset in Section 5.2.2.

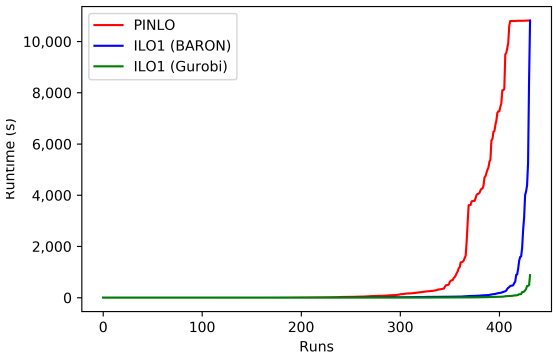
**WPI Dataset**

We conduct another experiment on a many-to-one stable matching application. The similar dataset presented in Chapter 4 is obtained from the Global Experience Office (GEO) at Worcester Polytechnic Institute (WPI) where each year over a thousand undergraduate students applying to over fifty of the esteemed Interactive Qualifying Project (IQP) off-campus project centers around the world. WPI students go through a competitive selection process, followed by rigorous cultural preparation. Some project center directors have interest in creating certain compositions of students through the concept of a *cohort*, which is a group of students having one or more particular features in common, such as academic major, language skill, or gender. For example, some project center directors may seek a minimum number of students with a particular language skill, some may seek to balance diversity, and still others may request a specific skill from a list of student majors. The concept of cohorts is incorporated in the model formulation (4.2) via goal programming techniques that penalize quadratic deviation from cohort targets with the computational results presented in Section 5.2.3.

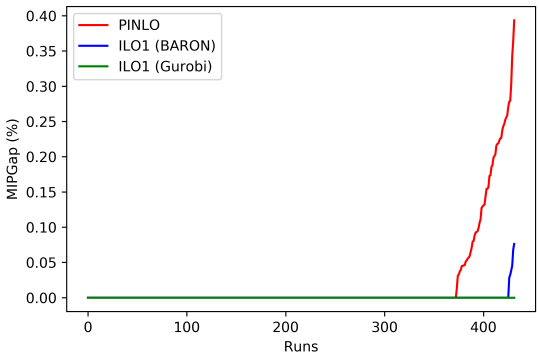
### 5.2.2 Experimental Results on Synthetic Dataset

We compare the performance of solving PINLO formulation (5.1) using BARON, ILO formulation (`ILO1`) using BARON, and ILO formulations (`ILO1`) and (`ILO2`) using Gurobi. First, we level the playing field of (5.1) formulation and (`ILO1`) formulation by comparing their results from BARON. Fig. 5.2 shows that (`ILO1`) outperforms (5.1) with respect to the average runtimes and MIP gaps over each run of (`ILO1`) (blue lines) as it results in more runs with lower runtime and MIP gap than (5.1) (red lines). When we solve ILO using Gurobi (green lines), all runs find optimal solutions with 0% MIP gaps with a mean

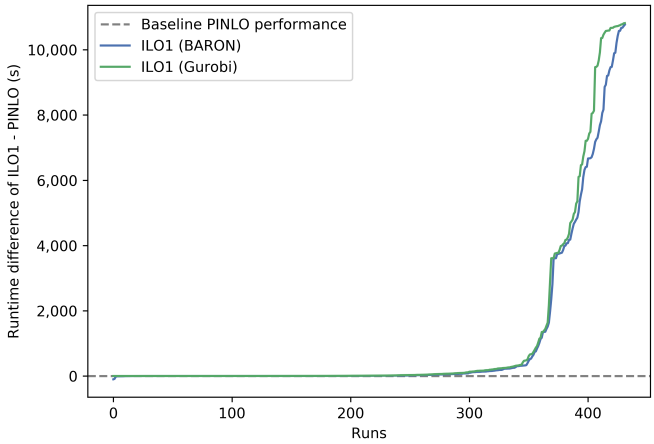
value of 14.6 seconds and the maximum (mean) runtime less than 900 seconds for all runs. Fig. 5.2c reveals that when comparing (ILO1) solved via either BARON or Gurobi versus PINLO solved via BARON, that over 100 runs result in notably faster computational performance (positive runtime difference) when comparing run-by-run differences in runtime. Specifically, by considering only instances with average runtime difference of ILO and PINLO strictly greater or less than 1% of the 3-hour time limit (or 108 seconds), ILO solved via BARON outperforms PINLO in 130 runs, while ILO solved via Gurobi outputforms PINLO in 134 runs, and for no runs did PINLO outperform either ILO.




(a) Average runtimes.




(b) Average MIP gaps.

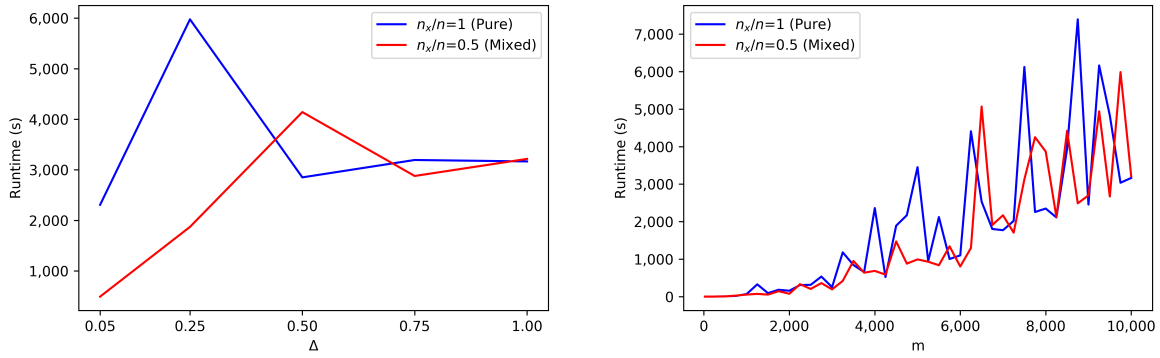



(c) Run-by-run comparison of runtime performance gain in seconds of (ILO1) over PINLO when solving (ILO1) via BARON and Gurobi, where positive runtime gain indicates faster performance of (ILO1) and the run-by-run performance gain lines of ILO1 are plotted after sorting in the ascending order.

Figure 5.2: Comparative results of 432 synthetic runs each averaging over three replicates for PINLO solved using BARON (red line) and (ILO1) solved using BARON (blue line) and using Gurobi (green line).

To view a bigger picture of ILO performance, we fix every parameter at their largest

value and increase $m$ up to 10,000, resulting in a maximum model size of 20,000 variables with $\alpha = 2$. We first vary the density of nonzero elements in the objective function by varying $\Delta \in \{0.05, 0.25, 0.5, 0.75, 1\}$. Fig. 5.3a show that (ILO1) finds optimal solutions to all instances of both mixed-integer (red line) and pure integer (blue line) types, with a maximum average runtime of 4,000 and 6,000 seconds, respectively. We also vary the number of constraints $m$ by adding 250 constraints to the maximum size up to 10,000 constraints, resulting in $m \in \{25, 50, 75, 100, 150, 250, 500, \ldots, 9{,}750, 10{,}000\}$. Fig. 5.3b shows that (ILO1) finds optimal solutions to all instances of both mixed-integer (red line) and pure integer (blue line) types, with a maximum average runtime of 6,000 and 7,400 seconds, respectively.

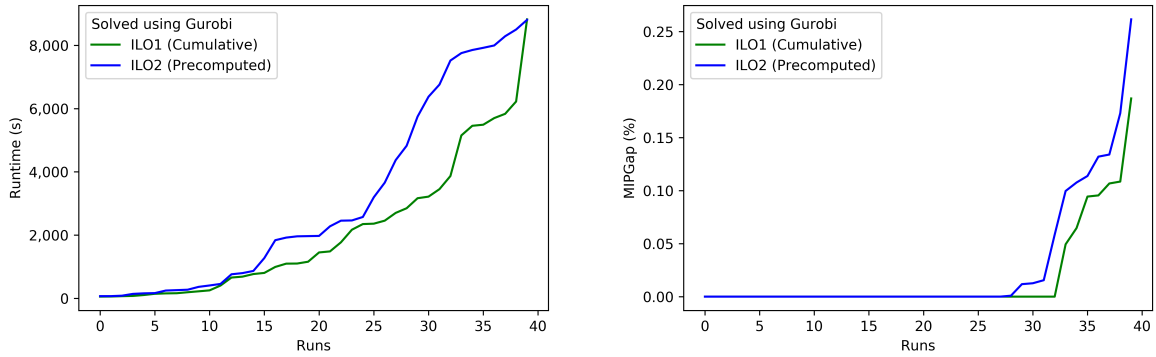



(a) Varying $\Delta$.

(b) Varying $m$.

Figure 5.3: Computational performance of (ILO1) solved using Gurobi on problems instances with up to 10,000 constraints and 20,000 variables where red and blue lines represent mixed-integer and pure integer instances, respectively.

As our limited experiments show that the computational performance of solving (ILO1) via both BARON and Gurobi is superior to solving PINLO with BARON and is computationally tractable for solving large problems of size 10,000 constraints and 20,000 variables, we next focus our investigations on the computational performance of (ILO1) and (ILO2). We fix every parameter at their largest value and vary the upper bound of integer variables $x^U$, the problem types $\frac{n_x}{n}$, and the number of constraints $m \in \{1{,}000, 2{,}000, \ldots, 10{,}000\}$, resulting in 40 instances. The results of each instance is averaged over three replicates. Fig. 5.4 plots the results from smaller to larger problem size. After a brief period where the computational performance of the two approaches appear similar, it becomes clear that (ILO1) outperforms (ILO2) for larger problem instances. Additionally, Fig. 5.4b shows that (ILO1) results in more optimal instances than (ILO2). We next demonstrate the use of our (ILO1) in solving a real-world polynomial integer optimization.




(a) Average runtimes.

(b) Average MIP gaps.

Figure 5.4: Comparing the Computational performance of (`ILO1`) and (`ILO2`) solved using Gurobi on problems instances with up to 10,000 constraints and 20,000 variables.

### 5.2.3 Experimental Results on WPI Dataset

We compare the performance of PINLO formulation (5.1) and ILO formulation (`ILO1`) on the *Quadratic Model Formulation of Stable Student Project Center (SPC) Allocation with Cohorts* presented in Chapter 4.4 across three academic years of the WPI dataset. The model formulation (4.2) produces a squared deviation penalty weighted by overassignment weight $\overline{w}_{aip}$ and underassignment weight $\underline{w}_{aip}$. For the purpose of experimentation, the cohort construction is selected so that all project center directors request at least 20% or more of the students to be female and at least 20% or more to be computer science (CS) majors. Thus, objective function (4.2a) is penalized via underassignment only when the total number of underassigned female students or underassigned CS students is below $\lceil 0.2 \times c_p \rceil$. The model is reproduced using a classical stability representation proposed by Baïou and Balinski [91] and other three stability representations: (`SPC1`), (`SPC2`), and (`VVM`), presented in Chapter 4.3.

Solving stable many-to-one matching with incomplete preference lists and ties is a challenging $NP$-hard optimization problem. Even though adding the construction of cohorts to the optimization formulation introduces additional complexity, ILO performs slightly better than the polynomial integer quadratic optimization (PIQO) in all three years of WPI dataset in terms of objective function quality, and gap. Table 5.2 shows that while the quality of the placements of PIQO and ILO formulations are roughly the same across all years and all stability representation techniques, ILO models result in fewer underassignments of females and CS students. Their differing performance is prominent in academic year 2018–2019 as all ILO models find best incumbent and optimal solutions in less than three hours, compared to PIQO models that spend a day to still prove optimality of the solution found. Our ILO formulation (`ILO1`) improves the computational

| **Year** | 2017-1018 | | 2018-1019 | | 2019-2020 | |
|---|---|---|---|---|---|---|
| **BBT** | **Cohort Type** | | **Cohort Type** | | **Cohort Type** | |
| | **PIQO** | **ILO** | **PIQO** | **ILO** | **PIQO** | **ILO** |
| Objective Value | 926.84 | 927.84 | 927.92 | 927.92 | 1,106.60 | 1,106.63 |
| Best Bound | 928.87 | 928.86 | 927.92 | 927.92 | 1,126.74 | 1,126.68 |
| MIP Gap (%) | 0.2198 | 0.1093 | 0.0005 | 0 | 1.82 | 1.8121 |
| Presolved Model Size | | | | | | |
| #Constraints (Rows) | 11,038 | 14,502 | 7,697 | 11,160 | 8,424 | 12,669 |
| #Variables (Columns) | 12,515 | 14,405 | 9,063 | 10,964 | 10,478 | 12,806 |
| # Nonzeros | 2,032,512 | 2,033,386 | 1,082,641 | 1,094,728 | 1,347,772 | 1,370,400 |
| Model Density | 0.0147 | 0.0097 | 0.0155 | 0.0089 | 0.0153 | 0.0084 |
| Tier-1 Placements | 863 | 848 | 919 | 919 | 965 | 961 |
| Tier-2 Placements | 63 | 79 | 8 | 8 | 141 | 145 |
| Unassignments | 2 | 1 | 0 | 0 | 20 | 20 |
| Underassigned Females | 3 | 0 | 0 | 0 | 21 | 18 |
| Underassigned CS | 67 | 66 | 56 | 56 | 96 | 96 |
| Time to Build Model (s) | 49.98 | 61.42 | 45.63 | 49.61 | 92.78 | 87.60 |
| Incumbent Time (s) | 610 | 83,432 | 80,548 | 1,300 | 19,544 | 7,566 |
| Run Time (s) | 86,400 | 86,400 | 86,400 | 1,301 | 86,400 | 86,400 |

Table 5.2: Comparing quadratic and linear stability formulations with cohorts on WPI datasets across three years using the BBT stability representation.

| **Year** | 2017-1018 | | 2018-1019 | | 2019-2020 | |
|---|---|---|---|---|---|---|
| **SPC1** | **Cohort Type** | | **Cohort Type** | | **Cohort Type** | |
| | **PIQO** | **ILO** | **PIQO** | **ILO** | **PIQO** | **ILO** |
| Objective Value | 927.84 | 927.84 | 927.91 | 927.92 | 1,106.68 | 1,108.67 |
| Best Bound | 927.84 | 928.84 | 927.92 | 927.92 | 1,126.74 | 1,126.73 |
| MIP Gap (%) | 0 | 0.1074 | 0.0006 | 0 | 1.8124 | 1.6291 |
| Presolved Model Size | | | | | | |
| #Constraints (Rows) | 13,931 | 17,425 | 12,075 | 15,561 | 12,711 | 17,013 |
| #Variables (Columns) | 18,376 | 20,306 | 15,474 | 17,418 | 16,688 | 19,062 |
| # Nonzeros | 166,406 | 175,154 | 114,416 | 123,126 | 93,044 | 103,804 |
| Model Density | 0.0007 | 0.0005 | 0.0006 | 0.0005 | 0.0004 | 0.0003 |
| Tier-1 Placements | 851 | 850 | 920 | 919 | 979 | 974 |
| Tier-2 Placements | **74** | 77 | 0 | 8 | 118 | 134 |
| Unassignments | 1 | 1 | 0 | 0 | 20 | 18 |
| Underassigned Females | 1 | 1 | 1 | 0 | 20 | 18 |
| Underassigned CS | 66 | 66 | 56 | 56 | 87 | 90 |
| Time to Build Model (s) | 129.85 | 121.76 | 98.26 | 137.69 | 234.86 | 216.23 |
| Incumbent Time (s) | 82,888 | 33,961 | 12,275 | 3,148 | 60,416 | 85,305 |
| Run Time (s) | 83,302 | 86,400 | 86,400 | 3,169 | 86,400 | 86,400 |

Table 5.3: Comparing quadratic and linear stability formulations with cohorts on WPI datasets across three years using the SPC1 stability representation.

| **Year** | 2017-1018 | | 2018-1019 | | 2019-2020 | |
|---|---|---|---|---|---|---|
| **SPC2** | **Cohort Type** | | **Cohort Type** | | **Cohort Type** | |
| | **PIQO** | **ILO** | **PIQO** | **ILO** | **PIQO** | **ILO** |
| Objective Value | 926.84 | 926.85 | 927.90 | 927.91 | 1,107.59 | 1,107.68 |
| Best Bound | 928.87 | 928.86 | 927.92 | 927.92 | 1,126.78 | 1,126.76 |
| MIP Gap (%) | 0.2198 | 0.2176 | 0.0023 | 0.0005 | 1.7327 | 1.7227 |
| Presolved Model Size | | | | | | |
| #Constraints (Rows) | 20,239 | 23,733 | 13,826 | 19,381 | 16,418 | 20,748 |
| #Variables (Columns) | 24,406 | 26,336 | 16,183 | 20,197 | 18,412 | 20,814 |
| # Nonzeros | 102,863 | 111,637 | 76,823 | 89,873 | 84,629 | 95,448 |
| Model Density | 0.0002 | 0.0002 | 0.0003 | 0.0002 | 0.0003 | 0.0002 |
| Tier-1 Placements | 860 | 861 | 927 | 922 | 975 | 956 |
| Tier-2 Placements | 66 | 65 | 0 | 5 | 132 | 151 |
| Unassignments | 2 | 2 | 0 | 0 | 19 | 19 |
| Underassigned Females | 2 | 0 | 3 | 2 | 20 | 18 |
| Underassigned CS | 67 | 67 | 56 | 56 | 103 | 86 |
| Time to Build Model (s) | 18.48 | 19.37 | 16.62 | 19.25 | 37.29 | 36.41 |
| Incumbent Time (s) | 12,019 | 28,364 | 24,118 | 8,942 | 707 | 86,400 |
| Run Time (s) | 86,400 | 86,400 | 86,400 | 86,400 | 86,400 | 86,400 |

Table 5.4: Comparing quadratic and linear stability formulations with cohorts on WPI datasets across three years using the SPC2 stability representation.

| **Year** | 2017-1018 | | 2018-1019 | | 2019-2020 | |
|---|---|---|---|---|---|---|
| **VVM** | **Cohort Type** | | **Cohort Type** | | **Cohort Type** | |
| | **PIQO** | **ILO** | **PIQO** | **ILO** | **PIQO** | **ILO** |
| Objective Value | 927.82 | 927.84 | 927.92 | 927.92 | 1,107.60 | 1,107.65 |
| Best Bound | 928.87 | 928.84 | 927.92 | 927.92 | 1,126.66 | 1,124.68 |
| MIP Gap (%) | 0.1133 | 0.1081 | 0.0005 | 0 | 1.7203 | 1.537 |
| Presolved Model Size | | | | | | |
| #Constraints (Rows) | 15,676 | 19,431 | 12,224 | 15,683 | 13,744 | 18,046 |
| #Variables (Columns) | 15,401 | 17,329 | 12,231 | 14,152 | 13,910 | 16,284 |
| # Nonzeros | 1,903,312 | 1,923,961 | 1,320,022 | 1,347,819 | 1,448,615 | 1,478,898 |
| Model Density | 0.0079 | 0.0057 | 0.0088 | 0.0061 | 0.0076 | 0.0050 |
| Tier-1 Placements | 850 | 847 | 921 | 919 | 946 | 951 |
| Tier-2 Placements | 77 | 80 | 6 | 8 | 161 | 156 |
| Unassignments | 1 | 1 | 0 | 0 | 19 | 19 |
| Underassigned Females | 4 | 9 | 1 | 23 | 20 | 19 |
| Underassigned CS | 66 | 66 | 56 | 56 | 97 | 89 |
| Time to Build Model (s) | 46.47 | 44.33 | 45.76 | 44.50 | 78.17 | 76.73 |
| Incumbent Time (s) | 133 | 83,053 | 6,463 | 815 | 38,731 | 31,724 |
| Run Time (s) | 86,400 | 86,400 | 86,400 | 817 | 86,400 | 86,400 |

Table 5.5: Comparing quadratic and linear stability formulations with cohorts on WPI datasets across three years using the VVM stability representation.

performance of solving the quadratic stable SPC allocation with cohorts formulation (4.1) as it performs slightly better than PIQO formulation (5.1) in terms of optimal objective function values and MIP gaps. Furthermore, the quality of the match outcomes in terms of underassigned females and CS students of (`ILO1`) is superior to PIQO formulation (5.1). As a result, our limited experiments on the synthetic and real datasets demonstrate that our ILO formulation (`ILO1`) is promising for solving integer linear optimization problems with a polynomial objective function.

## 5.3 Conclusions and Future Work

We study PINLO with mixed-integer polynomial objective function and linear constraints, which covers problem MIQP and related problem classes with separable polynomial expressions in the objective function. We theoretically derive the finite summation identity motivated from [92] that is used to reformulate PINLO problems to ILO problems via cumulative weighting. Our novel linearization advances linearization techniques for a large class of integer nonlinear optimization problems, converting them to ILO problems that we demonstrate are more computationally tractable. Additionally, we evaluate the performance of ILO on a synthetic dataset and a stable many-to-one matching application with cohort construction.

Our limited computational experiments on the synthetic dataset show that ILO outperforms PINLO when we solve using BARON via NEOS server. ILO is even faster using Gurobi. Our reformulation is promising in applying to real-world applications. As we apply ILO to solve a many-to-one stable matching application with cohort construction, ILO improves the match outcomes and computational runtime over three years of WPI dataset.

We believe that the reformulation of a class of PINLO in this study will benefit other applications. Future extensions may study how to apply the reformulation to other nonlinear functions in a similar manner using other algebraic identities, as well as applying our reformulation to polynomial constraints of the same class.

# Chapter 6

# Conclusions

This dissertation features the theoretical results and applications of mixed-integer nonlinear optimization (MINLO). MINLO appears in a wide range of fields, including chemical engineering, finance, and manufacturing. MINLO is a challenging problem due to nonconvex, nonlinear, and exponential increase in computational complexity as the dimension increases. Although algorithmic development and computing power incredibly advance the potential of optimization solvers during the past several decades, theoretical advancements will still have a major impact. We present multiple mathematical formulations of MINLO for matching applications as well as the reformulation of MINLO to improve computational tractability. Additionally, we conduct experiments on synthetic and real datasets to demonstrate the use of our formulations.

We use MINLO to formulate and solve maximum a-postiori estimation (MAP) for Gaussian Mixture Models (GMM) to assign data points into the right clusters in Chapter 2. We solve MINLO using Branch-and-Bound algorithm which can find global optima and near optima, and provide strict upper and lower bounds on the global optima. Our formulation is also easy to interpret statistically and adaptable to various prior distributions and constraints. We further reformulate MINLO to a mixed-integer quadratic optimization (MIQO) using piecewise linear functions to approximate the nonlinear objective function component and McCormick envelopes to linearize bilinear components of MAP. We conduct computational experiments on three well-known UCI datasets and a clinical dataset from the UNC MicroArray Database. Our analysis shows that MIQO yields better result when sample size is large as well as the performance can be significantly improved via incorporating prior knowledge through hard constraints.

In Chapter 3, we formulate MINLO model to automatically process tickets in IT services. Our automatic system first classifies tickets by using TF-IDF to extract keywords and modeling with logistic regression or neural network techniques resulting in over 78% accuracy. The system's precision can increase overtime as the learning set gains more

tickets and the system matures. Then, the system assigns the right tickets to the right technicians using MINLO model that considers required skills and time to fulfill a given ticket and technician's skillset, cost, and availability. Our decision support system can be readily adjusted and easily implemented in other applications. We hope to conserve resources, and increase efficiency in classifying and assigning tickets.

The use of optimization-based approaches has become more popular for solving many-to-one matching problems as they can be designed to maximize efficiency, ensure stability, and readily accommodate side constraints and incomplete preference lists with ties. In Chapter 4, we introduce new stability representations that can outperform the state-of-the-art in some settings. We theoretically show their correctness and create algorithms to accelerate the performance of our stability representations. We incorporate the concept of cohort (team) construction via goal programming that penalizes the deviation from desired cohort targets in a quadratic manner. Our model lexicographically optimize each component in objective functions in strict ordering. Our limited experiments over synthetic and real students to project centers allocation datasets show that our stability representation (`SPC2`) is promising for typical real-world applications where sufficient seats exist for applicants and our stability representation (`VVM`) exhibits solid performance overall.

In Chapter 5 we study a broad class of polynomial integer nonlinear optimization (PINLO) problems with mixed-integer polynomial objective function, linear constraints, and bounded and separable integer variables. Our PINLO covers MIQP and related problem classes that can be viewed as a generalization of integer linear optimization (ILO) with separable polynomial expressions in the objective function. We introduce two PINLO-to-ILO reformulation approaches deriving from different integer polynomial expressions. We theoretically derive the finite summation identity used to reformulate PINLO-to-ILO problems via cumulative weighting. Our computational experiments on synthetic and real datasets demonstrate that our novel linearization is more computationally tractable for solving PINLO problems with 10,000 constraints and 20,000 variables using state-of-the-art commercial solvers.

Our work contributes new theoretical concepts and ideas to MINLO problems. We demonstrate the use of our formulations as well as analyze the computational results. We hope that our formulations can benefit other applications. Future work consists of incorporating our introduced formulations into real applications, conducting further analysis, or applying our work to other MINLO applications.

# References


[1] Dimitris Bertsimas, Angela King, and Rahul Mazumder. Best subset selection via a modern optimization lens. *The Annals of Statistics*, 44(2):813–852, 2016. ISSN 00905364. URL `http://www.jstor.org/stable/43818629`.

[2] Juan Pablo Vielma, Shabbir Ahmed, and George Nemhauser. Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions. *Operations Research*, 58(2):303–315, 2010.

[3] Garth P McCormick. Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems. *Mathematical programming*, 10(1):147–175, 1976.

[4] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

[5] Erwin Kalvelagen. some MINLP solution algorithms, 2003. URL `http://amsterdamoptimization.com/pdf/minlp.pdf`.

[6] Patrick Flaherty, Pitchaya Wiratchotisatian, Ji Ah Lee, Zhou Tang, and Andrew C. Trapp. MAP clustering under the Gaussian mixture model via mixed integer nonlinear optimization, 2020.

[7] Hari Bandi, Dimitris Bertsimas, and Rahul Mazumder. Learning a mixture of Gaussians via mixed-integer optimization. *INFORMS Journal on Optimization*, 1(3):221–240, 2019. doi: 10.1287/ijoo.2018.0009. URL `https://doi.org/10.1287/ijoo.2018.0009`.

[8] Jan Van Bon, Arjen De Jong, Axel Kolthof, Mike Pieper, Ruby Tjassing, Annelies Van Der Veen, and Tieneke Verheijen. *Foundations of IT Service Management Based on ITIL®*, volume 3. Van Haren, 2008.

[9] Christopher J Chagnon, Andrew C Trapp, and Soussan Djamasbi. Creating a decision support system for service classification and assignment through optimization. 2017.

[10] Jeff Rumburg. Metric of the month: Percent resolved level 1 capable. 2011.

[11] Atila Abdulkadiroğlu and Tayfun Sönmez. School choice: A mechanism design approach. *American economic review*, 93(3):729–747, 2003.

[12] Atila Abdulkadiroğlu, Parag Pathak, Alvin E Roth, and Tayfun Sönmez. Changing the Boston school choice mechanism. Technical report, National Bureau of Economic Research, 2006.

[13] Fuhito Kojima and M Utku Ünver. The Boston school-choice mechanism: An axiomatic approach. *Economic Theory*, 55(3):515–544, 2014.

[14] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[15] Alvin E Roth and Marilda Sotomayor. The college admissions problem revisited. *Econometrica: Journal of the Econometric Society*, pages 559–570, 1989.

[16] Alvin E Roth and Marilda Sotomayor. Two-sided matching. *Handbook of game theory with economic applications*, 1:485–541, 1992.

[17] Alvin E Roth. On the allocation of residents to rural hospitals: A general property of two-sided matching markets. *Econometrica: Journal of the Econometric Society*, pages 425–427, 1986.

[18] Maxence Delorme, Sergio García, Jacek Gondzio, Jörg Kalcsics, David Manlove, and William Pettersson. Mathematical models for stable matching problems with ties and incomplete lists. *European Journal of Operational Research*, 277(2):426–441, 2019. ISSN 0377–2217.

[19] David Delacrétaz, Scott Duke Kominers, Alexander Teytelboym, et al. Matching mechanisms for refugee resettlement. Technical report, University of Oxford, 2019. URL `http://t8el.com/wp-content/uploads/2019/12/DKT-MMRR-Dec2019.pdf`.

[20] Narges Ahani, Tommy Andersson, Alessandro Martinello, Alexander Teytelboym, and Andrew C. Trapp. Placement Optimization in Refugee Resettlement. *forthcoming in Operations Research*, 2021.

[21] José Alcalde and Salvador Barberà. Top dominance and the possibility of strategy-proof stable solutions to matching problems. *Economic theory*, 4(3):417–435, 1994.

[22] David F Manlove, Gregg O'Malley, Patrick Prosser, and Chris Unsworth. A constraint programming approach to the hospitals/residents problem. In *International*

*Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*, pages 155–170. Springer, 2007.

[23] Kolos Csaba Ágoston, Péter Biró, and Iain McBride. Integer programming methods for special college admissions problems. *Journal of Combinatorial Optimization*, 32 (4):1371–1399, 2016.

[24] Natsumi Shimada, Natsuki Yamazaki, and Yuichi Takano. Multi-objective optimization models for many-to-one matching problems. *Journal of Information Processing*, 28:406–412, 2020. doi: 10.2197/ipsjjip.28.406.

[25] Alvin E Roth. The college admissions problem is not equivalent to the marriage problem. *Journal of economic Theory*, 36(2):277–288, 1985.

[26] Tayfun Sonmez. Manipulation via Capacities in Two-Sided Matching Markets. *Journal of Economic Theory*, 77(1):197–204, November 1997. URL `https://ideas.repec.org/a/eee/jetheo/v77y1997i1p197-204.html`.

[27] Alvin E Roth. The economics of matching: Stability and incentives. *Mathematics of operations research*, 7(4):617–628, 1982.

[28] Lloyd Shapley and Herbert Scarf. On cores and indivisibility. *Journal of mathematical economics*, 1(1):23–37, 1974.

[29] John H. Vande Vate. Linear programming brings marital bliss. *Operations Research Letters*, 8(3):147–153, 1989.

[30] Alvin E Roth, Uriel G Rothblum, and John H Vande Vate. Stable matchings, optimal assignments, and linear programming. *Mathematics of operations research*, 18(4): 803–828, 1993.

[31] Mourad Baïou and Michel Balinski. The stable admissions polytope. *Mathematical programming*, 87(3):427–439, 2000.

[32] Augustine Kwanashie and David F Manlove. An integer programming approach to the hospitals/residents problem with ties. In *Operations Research Proceedings 2013*, pages 263–269. Springer, 2014.

[33] Virginia Hogan. *Matching In Groups: A Theoretical and Empirical Study*. PhD thesis, Department of Economics, Stanford University, 2013.

[34] Péter Biró, David F Manlove, and Iain McBride. The hospitals/residents problem with couples: Complexity and integer programming models. In *International Symposium on Experimental Algorithms*, pages 10–21. Springer, 2014.

[35] Oliver Hinder. The stable matching linear program and an approximate rural hospital theorem with couples. In *Proceedings of WINE*, volume 15, page 433, 2015.

[36] Joanna Drummond, Andrew Perrault, and Fahiem Bacchus. Sat is an effective and complete method for solving stable matching problems with couples. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[37] David F Manlove, Iain McBride, and James Trimble. "almost-stable" matchings in the hospitals/residents problem with couples. *Constraints*, 22(1):50–72, 2017.

[38] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual. Version 9.1, 2020. URL `http://www.gurobi.com`.

[39] MINLPLib (2021) Mixed-integer nonlinear programming library, 2021. URL `http://www.minlplib.org/`. last accessed on August 10, 2021.

[40] Duan Li and Xiaoling Sun. *Nonlinear integer programming*, volume 84. Springer Science & Business Media, 2006.

[41] Christodoulos A Floudas and Viswanathan Visweswaran. Quadratic optimization. In *Handbook of global optimization*, pages 217–269. Springer, 1995.

[42] Kurt M Bretthauer and Bala Shetty. The nonlinear resource allocation problem. *Operations research*, 43(4):670–683, 1995.

[43] Kurt M Bretthauer and Bala Shetty. The nonlinear knapsack problem–algorithms and applications. *European Journal of Operational Research*, 138(3):459–472, 2002.

[44] Kurt M Bretthauer and Bala Shetty. A pegging algorithm for the nonlinear resource allocation problem. *Computers & Operations Research*, 29(5):505–527, 2002.

[45] Hanan Luss and Shiv K Gupta. Allocation of effort resources among competing activities. *Operations Research*, 23(2):360–366, 1975.

[46] Soren S Nielsen and Stavros A Zenios. A massively parallel algorithm for nonlinear stochastic network problems. *Operations Research*, 41(2):319–337, 1993.

[47] Bala Shetty and R Muthukrishnan. A parallel projection for the multicommodity network model. *Journal of the Operational Research Society*, 41(9):837–842, 1990.

[48] H Martin Weingartner. Capital budgeting of interrelated projects: survey and synthesis. *Management Science*, 12(7):485–516, 1966.

[49] Giora Hanoch and Haim Levy. Efficient portfolio selection with quadratic and cubic utility. *The Journal of Business*, 43(2):181–189, 1970.

[50] Claude Henin and Jérôme Doutriaux. A specialization of the convex simplex method to cubic programming. *Rivista di matematica per le scienze economiche e sociali*, 3 (2):61–72, 1980.

[51] G Edward Fox, Norman R Baker, and John L Bryant. Economic models for r and d project selection in the presence of project interactions. *Management science*, 30(7): 890–902, 1984.

[52] Paweł Regucki, Marek Lewkowicz, and Renata Krzyżyńska. Optimization of thermal-flow processes in a system of conjugate cooling towers. *Heat Transfer Engineering*, 41(22):1938–1948, 2020.

[53] Kayse Lee Maass, Vera Mann Hey Lo, Anna Weiss, and Mark S Daskin. Maximizing diversity in the engineering global leadership cultural families. *Interfaces*, 45(4): 293–304, 2015.

[54] Ailsa H Land and Alison G Doig. An automatic method for solving discrete programming problems. In *50 Years of Integer Programming 1958-2008*, pages 105–132. Springer, 2010.

[55] Sangbum Lee and Ignacio E Grossmann. A global optimization algorithm for non-convex generalized disjunctive programming and applications to process systems. *Computers & Chemical Engineering*, 25(11-12):1675–1697, 2001.

[56] Edward MB Smith and Constantinos C Pantelides. Global optimisation of nonconvex MINLPs. *Computers & Chemical Engineering*, 21:S791–S796, 1997.

[57] Pietro Belotti, Jon Lee, Leo Liberti, François Margot, and Andreas Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods & Software*, 24(4-5):597–634, 2009.

[58] Hong S Ryoo and Nikolaos V Sahinidis. Global optimization of nonconvex NLPs and MINLPs with applications in process design. *Computers & Chemical Engineering*, 19(5):551–566, 1995.

[59] Hong S Ryoo and Nikolaos V Sahinidis. A branch-and-reduce approach to global optimization. *Journal of global optimization*, 8(2):107–138, 1996.

[60] Mohit Tawarmalani and Nikolaos V Sahinidis. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Mathematical programming*, 99(3):563–591, 2004.

[61] Mohit Tawarmalani and Nikolaos V Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103(2):225–249, 2005.

[62] Ioannis P Androulakis, Costas D Maranas, and Christodoulos A Floudas. $\alpha$BB: A global optimization method for general constrained nonconvex problems. *Journal of Global Optimization*, 7(4):337–363, 1995.

[63] Claire S Adjiman, Stefan Dallwig, Christodoulos A Floudas, and Arnold Neumaier. A global optimization method, $\alpha$BB, for general twice-differentiable constrained NLPs—I. theoretical advances. *Computers & Chemical Engineering*, 22(9):1137–1158, 1998.

[64] Claire S Adjiman, Ioannis P Androulakis, and Christodoulos A Floudas. A global optimization method, $\alpha$BB, for general twice-differentiable constrained NLPs—II. implementation and computational results. *Computers & chemical engineering*, 22 (9):1159–1179, 1998.

[65] Claire S Adjiman, Ioannis P Androulakis, and Christodoulos A Floudas. Global optimization of mixed-integer nonlinear problems. *AIChE Journal*, 46(9):1769–1797, 2000.

[66] E.M.L. Beale and John JH Forrest. Global optimization using special ordered sets. *Mathematical Programming*, 10(1):52–69, 1976.

[67] E.M.L. Beale and J.A. Tomlin. Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. *OR*, 69 (447-454):99, 1970.

[68] E.M.L. Beale. Branch and bound methods for mathematical programming systems. In *Annals of Discrete Mathematics*, volume 5, pages 201–219. Elsevier, 1979.

[69] Björn Geißler, Alexander Martin, Antonio Morsi, and Lars Schewe. Using piecewise linear functions for solving MINLPs. In *Mixed integer nonlinear programming*, pages 287–314. Springer, 2012.

[70] Ahmet B Keha, Ismael R de Farias Jr, and George L Nemhauser. Models for representing piecewise linear cost functions. *Operations Research Letters*, 32(1):44–48, 2004.

[71] Ahmet B Keha, Ismael R de Farias Jr, and George L Nemhauser. A branch-and-cut algorithm without binary variables for nonconvex piecewise linear optimization. *Operations research*, 54(5):847–858, 2006.

[72] Sven Leyffer, Annick Sartenaer, and Emilie Wanufelle. Branch-and-refine for mixed-integer nonconvex global optimization. *Preprint ANL/MCS-P1547-0908, Mathematics and Computer Science Division, Argonne National Laboratory*, 39:40–78, 2008.

[73] Juan Pablo Vielma and George L Nemhauser. Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *Mathematical Programming*, 128(1):49–72, 2011.

[74] Harold P Benson. A finite algorithm for concave minimization over a polyhedron. *Naval Research Logistics Quarterly*, 32(1):165–177, 1985.

[75] Kurt M Bretthauer, A Victor Cabot, and MA Venkataramanan. An algorithm and new penalties for concave integer minimization over a polyhedron. *Naval Research Logistics (NRL)*, 41(3):435–454, 1994.

[76] Hoang Tuy. Concave programming under linear constraints. *Soviet Math.*, 5:1437–1440, 1964.

[77] Roy E Marsten and Thomas L Morin. A hybrid approach to discrete mathematical programming. *Mathematical Programming*, 14(1):21–40, 1978.

[78] Padmanaban Kesavan, Russell J Allgor, Edward P Gatzke, and Paul I Barton. Outer approximation algorithms for separable nonconvex mixed-integer nonlinear programs. *Mathematical Programming*, 100(3):517–535, 2004.

[79] Ramkumar Karuppiah and Ignacio E Grossmann. A Lagrangean based branch-and-cut algorithm for global optimization of nonconvex mixed-integer nonlinear programs with decomposable structures. *Journal of global optimization*, 41(2):163–186, 2008.

[80] Claudia D'Ambrosio, Jon Lee, and Andreas Wächter. An algorithmic framework for MINLP with separable non-convexity. In *Mixed Integer Nonlinear Programming*, pages 315–347. Springer, 2012.

[81] Samuel Burer and Adam N Letchford. Non-convex mixed-integer nonlinear programming: A survey. *Surveys in Operations Research and Management Science*, 17(2): 97–106, 2012.

[82] Christodoulos A Floudas. *Deterministic global optimization: theory, methods and applications*, volume 37. Springer Science & Business Media, 2013.

[83] Christodoulos A Floudas and Chrysanthos E Gounaris. A review of recent advances in global optimization. *Journal of Global Optimization*, 45(1):3, 2009.

[84] Reiner Horst and Hoang Tuy. *Global optimization: Deterministic approaches*. Springer Science & Business Media, 2013.

[85] N. V. Sahinidis. *BARON 21.1.13: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual, 2017.

[86] Donald E Knuth. Johann Faulhaber and sums of powers. *Mathematics of Computation*, 61(203):277–294, 1993.

[87] William Gropp and Jorge J. Moré. Optimization environments and the NEOS server. In Martin D. Buhman and Arieh Iserles, editors, *Approximation Theory and Optimization*, pages 167–182. Cambridge University Press, 1997.

[88] Joseph Czyzyk, Michael P. Mesnier, and Jorge J. Moré. The NEOS server. *IEEE Journal on Computational Science and Engineering*, 5(3):68–75, 1998.

[89] Elizabeth D. Dolan. The NEOS server 4.0 administrative guide. Technical Memorandum ANL/MCS-TM-250, Mathematics and Computer Science Division, Argonne National Laboratory, 2001.

[90] Giorgio Gallo, Peter L Hammer, and Bruno Simeone. Quadratic knapsack problems. In *Combinatorial optimization*, pages 132–149. Springer, 1980.

[91] Mourad Baïou and Michel Balinski. The stable allocation (or ordinal transportation) problem. *Mathematics of Operations Research*, 27(3):485–503, 2002.

[92] Kieren MacMillan and Jonathan Sondow. Proofs of power sum and binomial coefficient congruences via Pascal's identity. *The American Mathematical Monthly*, 118 (6):549–551, 2011.

[93] Laurence A Wolsey. *Integer programming*. John Wiley & Sons, 1998.

# Appendices

# Appendix A

# Theoretical Results Related to New Stability Representations (Chapter 4)

We present a sequence of theoretical results that demonstrate our proposed stability representations yield stable solutions. Recall that the many-to-one matching in this study is in the context of *HRT-W* in which the objective emphasizes maximizing the number of matches as well as weighted utilities for both sides of the matching, and the preference lists can be derived from ranking of corresponding weights assigned to each matching pair. Proposition 2, Theorem 1 and Theorem 2, respectively, show that, in the the context of *HRT-W*, the stability constraint sets (VVM), (SPC1), and (SPC2) in conjunction with (NWS) result in no blocking pairs associated with either justified envy or waste, resulting in stable solutions. Theorem 3 shows the equivalence of (BBT) and (VVM). Theorem 4 shows the equivalence of (SPC1), (SPC2), and (VVM). Finally, Theorem 5 shows that the recurrence relationship in Algorithms 1 and 2, respectively, preserves the stability constraint sets (SPC1) and (SPC2).

**Proposition 2** *The one-to-one stability constraint of [29] can be extended to the HRT-W context as follows:*

$$\sum_{i \prec_p s} x_{ip} \leq c_p \left(1 - \sum_{j \prec_s p} x_{sj}\right) \quad \forall\ s \in \mathcal{S},\ \forall\ p \in \mathcal{P}, \tag{VVM}$$

*and when combined with* (NWS)*, ensures a stable system with no justified envy or waste.*

**Proof.** We translate constraint set (VV) to the many-to-one context: if student $s$ is assigned to a project center less desirable than $p$ $\left(\sum_{j \succeq_{sp}} x_{sj} = 0\right)$, then project center $p$ may be filled with up to $c_p$ students that it prefers at least as much as $s$ $\left(\sum_{i \prec_p s} x_{ip} = 0\right)$.

We obtain $\sum_{i \prec_p s} x_{ip} \leq c_p \sum_{j \succeq_s p} x_{sj}$ as a result.

All students are assigned to either a project center in their list or to the virtual project center, thus $\sum_{j \prec_s p} x_{sj} + \sum_{j \succeq_s p} x_{sj} = 1$. By substituting $\sum_{j \succeq_s p} x_{sj}$, we obtain $\sum_{i \prec_p s} x_{ip} \leq c_p \left(1 - \sum_{j \prec_s p} x_{sj}\right)$ for each $s \in \mathcal{S}$ and $p \in \mathcal{P}$ which is equivalent to constraint set (VVM).

Constraint set (VVM) eliminates blocking pairs $(s, p)$ associated with justified envy in which project center $p$ prefers student $s$ to student $i$ in its assignment and, concurrently, student $s$ prefers project center $p$ to their match $j$. Taken in conjunction with system (NWS) that forbids waste, this ensures there are no blocking pairs associated with either justified envy or waste, and thereby a stable outcome for (VVM). ■

**Theorem 1** *A matching* $\boldsymbol{x} \in \{0,1\}^{|\mathcal{S}| \times |\mathcal{P}|}$ *is stable if and only if constraint sets* (SPC1) *and* (NWS) *are satisfied.*

**Proof.** First, we show that in the presence of (NWS), the outcomes of (SPC1) contain no blocking pair associated with waste. By contradiction, suppose (NWS) is violated, then there exists at least one empty seat at project center $p$ where student $i$ prefers $p$ to their match and $p$ prefers $i$ to an empty seat. Recall that (NWb) activates $z_p$ to 1 when there is at least one empty seat at project center $p$. When constraint set (NWa) is violated, the value on its left-hand side will be greater than zero, that is, $\sum_{i \in \mathcal{S}_p} \sum_{\substack{j \in \mathcal{P}: \\ j \prec_i p}} x_{ij} > 0$. Thus, there exists a blocking pair $(i, p)$ in which student $i$ is assigned to project center $j$ who prefers $p$ to $j$ and, concurrently, project center $p$ is willing to accept student $i$ instead of having an empty seat; hence the outcome is not stable.

In the presence of (NWS), it is then sufficient to show that a matching $\boldsymbol{x} \in \{0,1\}^{|\mathcal{S}| \times |\mathcal{P}|}$ is stable if and only if constraint set (SPC1) is satisfied, that is it forbids blocking pairs associated with justified envy.

**Sufficiency.** By contradiction, suppose (SPC1) is violated. This implies the existence of both $(s, p) \in \mathcal{S} \times \mathcal{P} : x_{sp} = 1$ and $(i, j) \in \mathcal{S} \times \mathcal{P} : x_{ij} = 1$ with $j \succ_s p, i \prec_j s$. Thus, $(s, j)$ forms a blocking pair associated with justified envy and matching $\boldsymbol{x} \in \{0,1\}^{|\mathcal{S}| \times |\mathcal{P}|}$ violates the notion of stability.

**Necessity.** Now, suppose a matching $\boldsymbol{x} \in \{0,1\}^{|\mathcal{S}| \times |\mathcal{P}|}$ is *not* stable. A blocking pair associated with a match $(s, p) \in \mathcal{S} \times \mathcal{P}$ is either $(s, j)$ or $(i, p)$. Without loss of generality, suppose a blocking pair is $(s, j)$ in which student $s$ is assigned to project center $p$ and student $i$ is assigned to project center $j$, but $j \succ_s p$ and $i \prec_j s$. This implies that $\exists (s, p) \in \mathcal{S} \times \mathcal{P} : x_{sp} = 1$ while $\sum_{\substack{j \in \mathcal{P}: \\ j \succ_s p}} \sum_{\substack{i \in \mathcal{S}: \\ i \prec_j s}} x_{ij} \geq 1$. Thus, (SPC1) is not satisfied. A similar argument exists for a blocking pair $(i, p)$ where $(s, p) \in \mathcal{S} \times \mathcal{P}$ and $(i, j) \in \mathcal{S} \times \mathcal{P}$,

but $p \succ_i j$ and $s \prec_p i$. ■

**Theorem 2** *A matching* $\boldsymbol{x} \in \{0,1\}^{|\mathcal{S}|\times|\mathcal{P}|}$ *is stable if and only if constraint sets* (SPC2) *and* (NWS) *are satisfied.*

**Proof.**

First, we show that in the presence of (NWS), the outcomes of (SPC2) contain no blocking pair associated with waste. This can be shown in a similar manner as in Theorem 1. In the presence of (NWS), it is now sufficient to show that a matching $\boldsymbol{x} \in \{0,1\}^{|\mathcal{S}|\times|\mathcal{P}|}$ is stable if and only if constraint sets (SPC2) is satisfied, that is it forbids blocking pairs associated with justified envy.

**Sufficiency.** By contradiction, suppose (SPC2) is violated. This implies the existence of both $(s,p) \in \mathcal{S}\times\mathcal{P} : x_{sp} = 1$ and $(i,j) \in \mathcal{S}\times\mathcal{P} : x_{ij} = 1$ with $i \succ_p s, j \prec_i p$. Thus, $(i,p)$ forms a blocking pair and matching $\boldsymbol{x} \in \{0,1\}^{|\mathcal{S}|\times|\mathcal{P}|}$ violates the notion of stability.

**Necessity.** Now suppose a matching $\boldsymbol{x} \in \{0,1\}^{|\mathcal{S}|\times|\mathcal{P}|}$ is *not* stable. A blocking pair associated with a match $(s,p) \in \mathcal{S}\times\mathcal{P}$ is either $(s,j)$ or $(i,p)$. Without loss of generality, suppose a blocking pair is $(i,p)$ in which student $s$ is assigned to project center $p$ and student $i$ is assigned to project center $j$, but $p \succ_i j$ and $s \prec_p i$. This implies that $\exists(s,p) \in \mathcal{S}\times\mathcal{P} : x_{sp} = 1$ while $\sum_{\substack{i\in\mathcal{S}:\\ s\prec_p i}} \sum_{\substack{j\in\mathcal{P}:\\ p\succ_i j}} x_{ij} \geq 1$. Thus, (SPC2) is not satisfied. A similar argument exists for a blocking pair $(s,j)$ where $(s,p) \in \mathcal{S}\times\mathcal{P}$ and $(i,j) \in \mathcal{S}\times\mathcal{P}$, but $i \prec_j s$ and $j \succ_s p$. ■

We now demonstrate first the equivalence of (BBT) and (VVM) in Theorem 3, followed by the equivalence of (SPC1), (SPC2), and (VVM) in Theorem 4.

**Theorem 3** *Stability constraint sets* (BBT) *and* (VVM) *when combined with* (NWS) *achieve equivalent stable outcomes.*

**Proof.** As (NWS) forbids blocking pairs associated to waste, we show that (VVM) which forbids blocking pairs associated to justified envy is equivalent to (BBT). For each $s \in \mathcal{S}$ and $p \in \mathcal{P}$:

$$
\begin{aligned}
& c_p x_{sp} + c_p \sum_{\substack{j\neq p:\\ j\succeq_s p}} x_{sj} + \sum_{\substack{i\neq s:\\ i\succeq_p s}} x_{ip} \geq c_p && \text{stability constraint (BBT)} \\
\iff \quad & c_p \sum_{j\succeq_s p} x_{sj} + \sum_{i\succeq_p s} x_{ip} \geq c_p && \text{stability constraint (HRT)} \\
\iff \quad & c_p(1 - \sum_{j\prec_s p} x_{sj}) \geq c_p - \sum_{i\succeq_p s} x_{ip} \geq \sum_{i\prec_p s} x_{ip} && \text{all students are matched} \\
\iff \quad & c_p(1 - \sum_{j\prec_s p} x_{sj}) \geq \sum_{i\prec_p s} x_{ip} && \text{stability constraint (VVM)}
\end{aligned}
$$

■

**Theorem 4** *Stability constraint sets* (SPC1) *and* (VVM) *when both combined with* (NWS) *achieve equivalent stable outcomes.*

**Proof.** Given that (NWS) is applied to both constraint sets, we prove by contradiction the equivalence of (SPC1) and (VVM) in which they forbids blocking pairs associated to justified envy.

Suppose (SPC1) holds, but (VVM) is violated. This implies that $\sum_{j \prec_{sp}} x_{sj} = 1$ and yet $\sum_{i \prec_{ps}} x_{ip} \geq 1$. Therefore, $\exists (s,j),(i,p) \in \mathcal{S} \times \mathcal{P} : x_{sj} = 1$ and $x_{ip} = 1$, where $j \prec_s p$ and $i \prec_p s$. Accordingly, $x_{sp}$ is a blocking pair associated with justified envy, and thus stability constraint set (SPC1) cannot hold.

Now, suppose (VVM) holds, but (SPC1) is violated. This implies that $\exists (s,j),(i,p) \in \mathcal{S} \times \mathcal{P} : x_{sp} = 1$ and $x_{ij} = 1$, where $j \succ_s p$ and $i \prec_j s$. Accordingly, $x_{sj}$ is a blocking pair associated with justified envy, and thus stability constraint set (VVM) cannot hold. ∎

The equivalence of (SPC2) and (VVM) can be shown in a similar manner, this proving the equivalence of (SPC1), (SPC2), and (VVM). Theorems 3 and 4 together imply the equivalence of (SPC1), (SPC2), (VVM), and (BBT). We now show that Algorithms 1 and 2 preserve stability constraint sets (SPC1) and (SPC2), respectively.

**Theorem 5** *The recurrence relation of the variables $\alpha_{sp}$ in Algorithm 1 preserves stability constraint sets* (SPC1).

**Proof.** First, rank order the project centers $p \in \mathcal{P}$ according to the preference list of student $s$ as $p^{(1)}, p^{(2)}, \ldots$; then, for any $s \in \mathcal{S}$ and $p^{(r)} \in \mathcal{P}$:

$$
\begin{aligned}
& \alpha_{sp^{(r)}} \leq \mathcal{M}_s \left(1 - x_{sp^{(r)}}\right) \\
\iff \quad & \alpha_{sp^{(r-1)}} + \beta_{sp^{(r-1)}} \leq \mathcal{M}_s \left(1 - x_{sp^{(r)}}\right) \\
\iff \quad & \alpha_{sp^{(r-2)}} + \textstyle\sum_{k=1}^{2} \beta_{sp^{(r-k)}} \leq \mathcal{M}_s \left(1 - x_{sp^{(r)}}\right) \\
\iff \quad & \cdots \\
\iff \quad & \alpha_{sp^{(1)}} + \textstyle\sum_{k=1}^{r-1} \beta_{sp^{(r-k)}} \leq \mathcal{M}_s \left(1 - x_{sp^{(r)}}\right) \\
\iff \quad & \textstyle\sum_{k=1}^{r-1} \beta_{sp^{(r-k)}} \leq \mathcal{M}_s \left(1 - x_{sp^{(r)}}\right) \quad \text{by definition } \alpha_{sp^{(1)}} = 0 \\
\iff \quad & \textstyle\sum_{k=1}^{r-1} \sum_{i \prec_{p^{(r-k)}} s} x_{ip^{(r-k)}} \leq \mathcal{M}_s \left(1 - x_{sp^{(r)}}\right) \quad \text{by definition } \beta_{sp^{(r)}} = \sum_{\substack{i \in \mathcal{S}: \\ i \prec_{p^{(r)}} s}} x_{ip^{(r)}} \\
\iff \quad & \sum_{\substack{j \in \mathcal{P}: \\ j \succ_s p^{(r)}}} \sum_{\substack{i \in \mathcal{S}: \\ i \prec_j s}} x_{ij} \leq \mathcal{M}_s \left(1 - x_{sp^{(r)}}\right) \quad \text{by substituting } \{j \in \mathcal{P} : j \succ_s p^{(r)}\}.
\end{aligned}
$$

∎

A similar proof shows that the recurrence relation of the variable $\alpha_{sp}$ in Algorithm 2 preserves stability constraint set (SPC2).

# Appendix B

# Summary of Limited Computational Testing (Chapter 4)

We computationally investigate some parameters to shed light on desirable configurations for our computational experiments. First, limited computational testing revealed that the constraint set (PW) tends to be outperformed by both of the aggregate versions (SPC1) and (SPC2). This was surprising given the common understanding that the performance of aggregate constraints is typically dominated by disaggregated constraints in terms of the linear programming relaxation solution quality [see, e.g., 93]. Algorithms 1 and 2 use recurrence relationships to construct stability constraint sets (SPC1) and (SPC2) over sorted preference lists. As a result, the summation of binary variables appearing in (SPC1) and (SPC2) is not composed of binary variables, but rather an integer variable that is set equal to the sum of the respective binary variables. Limited computational testing reveals that models with these integer variables that represent binary variable aggregations result in superior performance, over models with only binary variables.

Additionally, we performed a number of limited computational experiments to understand what parameters and settings resulted in the best performance. Specifically, we compare the performance between using Big-M conditions and Gurobi indicator constraints. Even though we reduce the magnitude of Big-M parameters that appear in stability constraint sets (SPC1) and (SPC2) by enumerating through binary variables $x_{ij}$ representing possible blocking pairs related to justified envy which locating on the left-hand side of the constraint sets.

# Appendix C

# An Example to Demonstrate the Generation of Constraint Sets via Algorithm 1 and Algorithm 2 (Chapter 4)

**Example 2** *example Suppose there are two project centers $p_1, p_2$, each with two seats, and three students $s_1, s_2, s_3$ with incomplete preference lists and ties. Preferences and priorities are shown in Figure C.1.*

$$
\begin{array}{ll}
s_1: (p_1) & p_1: (s_2, s_3), (s_1) \\
s_2: (p_1, p_2) & p_2: (s_1), (s_2), (s_3) \\
s_3: (p_2), (p_1) &
\end{array}
$$

Figure C.1: Illustrating SPC assignment with incomplete preference lists and ties of Example 2. Unranked project centers / students are unlisted, and entities ranked at the same level are enclosed within same brackets.

Algorithm 1 iterates over the preference of students, whereas Algorithm 2 iterates over the preference of project centers, both sorted in a descending order. While both algorithms are based on (`PW`), each results in a unique constraint set as shown in Table C.1. We now illustrate the recurrence relations of Algorithms 1 and 2 for constructing (`SPC1`) and (`SPC2`).

**Algorithm 1** Considering the sorted project centers of each student $s$ in a descending order, Algorithm 1 creates constraints to prevent any assignments of other students who each sorted project center $p$ prefers less than $s$. A different subset of these constraints is activated when student $s$ is assigned to each sorted project centers $p \in \mathcal{P}$.

From the preference order of $s_1$, as $s_1$ prefers being unmatched to being assigned to $p_2$, the order of project centers of $s_1$ is $(p_1), (p_0), (p_2)$. Because $p_1$ prefers no one less than $s_1$, this generates $\beta_{1,1} = 0$ and $\alpha_{1,1} = 0$. Next, $p_0$ prefers everyone equally generates $\beta_{1,0} = 0$ and $\alpha_{1,0} = \alpha_{1,1} + \beta_{1,1}$. Then, $p_2$ prefers $s_2$ and $s_3$ less than $s_1$ generates $\beta_{1,2} = x_{2,2} + x_{3,2}$ and $\alpha_{1,2} = \alpha_{1,0} + \beta_{1,0}$.

From the preference order of $s_2$ where $p_1$ and $p_2$ have the same ranking, $p_1$ prefers $s_1$ less than $s_2$ generates $\beta_{2,1} = x_{1,1}$ and $\alpha_{2,1} = 0$, whereas $p_2$ prefers $s_3$ less than $s_2$ generates $\beta_{2,2} = x_{3,2}$ and $\alpha_{2,2} = 0$. Then, $p_0$ generates $\beta_{2,0} = 0$ and $\alpha_{2,0} = \alpha_{2,2} + \beta_{2,2} + \beta_{2,1}$.

From the preference order of $s_3$, first, $p_2$ prefers no one less than $s_3$ generates $\beta_{3,2} = 0$ and $\alpha_{3,2} = 0$. Next, $p_1$ prefers $s_1$ less than $s_3$ generates $\beta_{3,1} = x_{1,1}$ and $\alpha_{3,1} = \alpha_{3,2} + \beta_{3,2}$. Then, $p_0$ generates $\beta_{3,0} = 0$ and $\alpha_{3,0} = \alpha_{3,1} + \beta_{3,1}$.

All in all, Algorithm 1 generates 9 constraints of the form $\alpha_{sp} \leq \mathcal{M}_s(1 - x_{sp})$, the last of which is $\alpha_{3,0} = \alpha_{3,1} + \beta_{3,1} = x_{1,1} \leq \mathcal{M}_s(1 - x_{3,0})$.

**Algorithm 2** Considering the sorted students of each project center $p$ in a descending order, Algorithm 2 creates constraints to prevent any assignments of each sorted student $s$ to any of other project centers where $s$ prefers less than $p$. A different subset of these constraints is activated when project center $p$ is assigned each sorted student $s \in \mathcal{S}$.

From the preference order of $p_1$ where $s_2$ and $s_3$ have the same ranking, $s_2$ prefers $p_0$ less than $p_1$ generates $\beta_{2,1} = x_{2,0}$ and $\alpha_{2,1} = 0$, whereas $s_3$ prefers $p_0$ less than $p_1$ generates $\beta_{3,1} = x_{3,0}$ and $\alpha_{3,1} = 0$. Next, $s_1$ prefers $p_0$ and $p_2$ less than $p_1$ generates $\beta_{1,1} = x_{1,0} + x_{1,2}$ and $\alpha_{1,1} = \alpha_{3,1} + \beta_{3,1} + \beta_{2,1}$.

From the preference order of $p_2$, first, $s_1$ prefers nowhere less than $p_2$ generates $\beta_{1,2} = 0$ and $\alpha_{1,2} = 0$. Next, $s_2$ prefers $p_0$ less than $p_2$ generates $\beta_{2,2} = x_{2,0}$ and $\alpha_{2,2} = \alpha_{1,2} + \beta_{1,2}$. Then, $s_3$ prefers $p_1$ and $p_0$ less than $p_2$ generates $\beta_{3,2} = x_{3,1} + x_{3,0}$ and $\alpha_{3,2} = \alpha_{2,2} + \beta_{2,2}$.

All in all, Algorithm 2 generates 6 constraints of the form $\alpha_{sp} \leq \mathcal{M}_p(1 - x_{sp})$, the last of which is $\alpha_{3,2} = \alpha_{2,2} + \beta_{2,2} = x_{2,0} \leq \mathcal{M}_s(1 - x_{3,2})$.

Note that we use Gurobi indicator constraints to model Big-M conditions in stability representations (`SPC1`) and (`SPC2`) as our limited experiment shows their superior performance over Big-M conditions. We summarize constraint sets generated by Algorithm 1 and Algorithm 2 in Table C.1.

| **Constraint generated by Algorithm 1** | **Constraint generated by Algorithm 2** |
|---|---|
| $\beta_{1,1} = 0$ | $\beta_{3,1} = x_{3,0}$ |
| $\alpha_{1,1} = 0$ | $\alpha_{3,1} = 0$ |
| $\beta_{1,0} = 0$ | $\beta_{2,1} = x_{2,0}$ |
| $\alpha_{1,0} = \beta_{1,1} + \alpha_{1,1}$ | $\alpha_{2,1} = 0$ |
| $\beta_{1,2} = x_{2,2} + x_{3,2}$ | $\beta_{1,1} = x_{1,2} + x_{1,0}$ |
| $\alpha_{1,2} = \beta_{1,0} + \alpha_{1,0}$ | $\alpha_{1,1} = \beta_{3,1} - \alpha_{3,1} - \beta_{2,1}$ |
| $\beta_{2,2} = x_{3,2}$ | $\beta_{1,2} = 0$ |
| $\alpha_{2,2} = 0$ | $\alpha_{1,2} = 0$ |
| $\beta_{2,1} = x_{1,1}$ | $\beta_{2,2} = x_{2,0}$ |
| $\alpha_{2,1} = 0$ | $\alpha_{2,2} = \beta_{1,2} - \alpha_{1,2}$ |
| $\beta_{2,0} = 0$ | $\beta_{3,2} = x_{3,1} - x_{3,0}$ |
| $\alpha_{2,0} = \beta_{2,2} + \alpha_{2,2} + \beta_{2,1}$ | $\alpha_{3,2} = \beta_{2,2} - \alpha_{2,2}$ |
| $\beta_{3,2} = 0$ | $\beta_{3,0} = 0$ |
| $\alpha_{3,2} = 0$ | $\alpha_{3,0} = 0$ |
| $\beta_{3,1} = x_{1,1}$ | $\beta_{2,0} = 0$ |
| $\alpha_{3,1} = \beta_{3,2} + \alpha_{3,2}$ | $\alpha_{2,0} = 0$ |
| $\beta_{3,0} = 0$ | $\beta_{1,0} = x_{1,2}$ |
| $\alpha_{3,0} = \beta_{3,1} + \alpha_{3,1}$ | $\alpha_{1,0} = 0$ |
| Building (`SPC1`) via indicator constraints | Building (`SPC2`) via indicator constraints |
| $x_{1,1} = 1 \to \alpha_{1,1} = 0$ | $x_{3,1} = 1 \to \alpha_{3,1} = 0$ |
| $x_{1,0} = 1 \to \alpha_{1,0} = 0$ | $x_{2,1} = 1 \to \alpha_{2,1} = 0$ |
| $x_{1,2} = 1 \to \alpha_{1,2} = 0$ | $x_{1,1} = 1 \to \alpha_{1,1} = 0$ |
| $x_{2,2} = 1 \to \alpha_{2,2} = 0$ | $x_{1,2} = 1 \to \alpha_{1,2} = 0$ |
| $x_{2,1} = 1 \to \alpha_{2,1} = 0$ | $x_{2,2} = 1 \to \alpha_{2,2} = 0$ |
| $x_{2,0} = 1 \to \alpha_{2,0} = 0$ | $x_{3,2} = 1 \to \alpha_{3,2} = 0$ |
| $x_{3,2} = 1 \to \alpha_{3,2} = 0$ | $x_{3,0} = 1 \to \alpha_{3,0} = 0$ |
| $x_{3,1} = 1 \to \alpha_{3,1} = 0$ | $x_{2,0} = 1 \to \alpha_{2,0} = 0$ |
| $x_{3,0} = 1 \to \alpha_{3,0} = 0$ | $x_{1,0} = 1 \to \alpha_{1,0} = 0$ |

Table C.1: The generation order and constraints created by Algorithm 1 and Algorithm 2 are different.

# Appendix D

# Generating Student Preference List Based on the Two Types of `popularity` Parameter (Chapter 4)

The preference lists in our datasets are incomplete. Unlike the studies of [32] and [18] where the data in the randomly generated instances appears to be generated with ties on the hospital side only, we allow ties on both student and project center sides, with a greater tie density in student preference lists. We note that in the generated test instances of [32], all hospitals in the preference list of each doctor rank the particular doctor in the same tier. Additionally, the distribution of doctor grades is controlled by a skewness parameter, in which a skewness value of $\kappa$ means that the most common doctor score is likely to occur $\kappa$ times more than the least common, whereas the distribution of our student preference list is controlled by the popularity of each project center, according to the following interpretation. In our synthetic datasets, we generate a discrete probability distribution for tier selection based on the popularity ranking of each project center, such that more popular project centers will have greater likelihood of being placed in the first tier of student preference lists.

The preference lists of students and project centers in Section 4.5.5 are generated to reflect the SPC matching in which student preferences are constructed in preference tiers where project centers in the same tier represent ties, and project center preferences are calculated based on a variety of observable student characteristics such as resume, transcript, language skill, and personal statement, with normalized values between 0 and 1.

The student preferences is constructed based on the project center popularity where its distribution can be uniform or nonuniform. A *uniform* popularity indicates a uniform popularity distribution across all project centers, where each project center is likely to be

selected by a student with a probability of $\frac{1}{|\mathcal{P}|}$. The chance that each project center will be selected in the first tier of student preference lists are equal. A *nonuniform* popularity indicates a nonuniform popularity distribution across all project centers, where more popular project centers are more likely to be selected to higher tier of student preference lists.

We define a discrete probability distribution of tier selection for project popularity rank $p \in \{1, 2, \dots\}$ as

$$\Pr(\text{rank} = p, \text{tier} = r) = \frac{|\mathcal{R}_p| - r + p}{|\mathcal{R}_p| \left(\frac{|\mathcal{R}_p|-1}{2} + p\right)} \quad \text{for preference tier } r \in \{1, \dots, |\mathcal{R}_p|\}.$$

For example, when $|\mathcal{R}_p| = 3$, the probability that the most popular project center (rank=1) is selected in the first, second, and third tier of student preference lists are $\frac{3}{6}$, $\frac{2}{6}$, and $\frac{1}{6}$, respectively; the probability that the second most popular project center (rank=2) is selected in the first, second, and third tier of student preference lists are $\frac{4}{9}$, $\frac{3}{9}$, and $\frac{2}{9}$, respectively; and so on. The scale difference between each project popularity rank can be adjusted by changing the scale of $p$ values.

The probability distribution of the tier selection of a more popular project center results in higher chance that it will be selected in higher tier. The random selection of project popularity and preference lists is implemented using Python function `choice` in `numpy.random` library. With the use of this function, students place each project center to a tier according to a list of probability values generated from the corresponding project popularity rank. We prohibit student preference lists with empty first tier. Incomplete preference lists occur when students do not place project centers in every tier. Since we work with cardinal preferences, a utility value of a project center placed in the $r^{\text{th}}$ tier is defined to be $\frac{1}{r}$ for $r = 1, \dots, |\mathcal{R}_p| - 1$, and zero if it is placed in the last tier, $r = |\mathcal{R}_p|$.

# Appendix E

# Selecting Coefficients of the Lexicographic Objective Functions (Chapter 4)

## E.1 Objective Coefficients for the Stable SPC Formulation (4.1) of the WPI Datasets

To study the effect of different stability representations, we apply formulation (4.1) with the utility $u_{sp}$ set to the summation of weighted student utility and weighted project center utility, $\gamma_1 q_{sp} + \gamma_2 k_{sp}$. We choose the values of $\gamma_1$ and $\gamma_2$ so that the objective function:

$$\text{maximize} \quad \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} x_{sp} + \gamma_1 \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} q_{sp}x_{sp} + \gamma_2 \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} k_{sp}x_{sp},$$

preemptively optimizes in the following order: i) maximize student placement, ii) maximize student utility, and iii) maximize project center utility.

First, we would like to ensure that the contribution to the objective function of total project center utility is strictly less than the minimum contribution of the student utility from placing any single student: $\gamma_2 \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} k_{sp}x_{sp} < \gamma_1 \min_{\{s\in\mathcal{S},p\in\mathcal{P}\}} q^+_{sp}$. Second, we would like to ensure that the contribution to the objective function of total student utility is strictly less than the minimum contribution of a placement of any single student: $\gamma_1 \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} q_{sp}x_{sp} < 1$. Finally, we would like to ensure that the contribution to the objective function of total student utility and total project center utility together is strictly less than the minimum contribution of a placement of any single student: $\gamma_1 \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} q_{sp}x_{sp} + \gamma_2 \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} k_{sp}x_{sp} < 1$. To do so, it is helpful to work backwards, so

that first the $\gamma_2$ component is determined in term of $\gamma_1$, followed by solving for the value of the $\gamma_1$ component.

As the result, we set the value of $\gamma_1$ to be the reciprocal of the minimum of the total capacity and the number of students, plus the smallest possible positive contribution from student utility, plus a small positive value, that is $\gamma_1 = \frac{1}{\min\limits_{\{s\in\mathcal{S},p\in\mathcal{P}\}} q^+_{sp} + \min\{\sum\limits_{p\in\mathcal{P}} c_p, |\mathcal{S}|\} + \varepsilon}$. We set the value of $\gamma_2$ to be the product of $\gamma_1$ and the ratio of the minimum student utility of a placement of any single student to the minimum of the total capacity and the number of students plus a small positive value, $\gamma_2 = \frac{\min\limits_{\{s\in\mathcal{S},p\in\mathcal{P}\}} q^+_{sp}}{\min\{\sum\limits_{p\in\mathcal{P}} c_p, |\mathcal{S}|\} + \varepsilon}\gamma_1$.

For the WPI dataset, the capacity is dropped to its observed level in every year, $\sum\limits_{p\in\mathcal{P}} c_p = |\mathcal{S}|$, and we prohibit the placement of students to unpreferred project centers in which $q_{sp} = 0$ via constraint (4.1d), which results in $\min\limits_{\{s\in\mathcal{S},p\in\mathcal{P}\}} q^+_{sp} = 0.5$. Hence, $\gamma_1 = \frac{1}{0.5 + \sum\limits_{p\in\mathcal{P}} c_p + \varepsilon}$ and $\gamma_2 = \frac{0.5}{\sum\limits_{p\in\mathcal{P}} c_p + \varepsilon}\frac{1}{0.5 + \sum\limits_{p\in\mathcal{P}} c_p + \varepsilon}$, where we set the value of $\varepsilon$ to be 1e$-$6.

## E.2 Coefficients for the Stable SPC with Cohorts Formulation (4.2) of the WPI Datasets

We choose the values of $\overline{w}_{aip}$ and $\underline{w}_{aip}$ so that the objective function:

$$\text{maximize} \quad \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} x_{sp} + \gamma_1 \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} q_{sp}x_{sp} + \gamma_2 \sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}} k_{sp}x_{sp} - \sum_{a\in\mathcal{A}}\sum_{i\in\mathcal{I}^a}\sum_{p\in\mathcal{P}} \overline{w}_{aip}\overline{y}^2_{aip} - \sum_{a\in\mathcal{A}}\sum_{i\in\mathcal{I}^a}\sum_{p\in\mathcal{P}} \underline{w}_{aip}\underline{y}^2_{aip},$$

preemptively optimizes in the order specified for the stable SPC formulation (4.1) of the WPI datasets and then minimizes penalty of the squared deviation from desired cohort target.

We ensure that the contribution to the objective function of the squared deviation penalty is strictly less than the minimum contribution of a placement of one student and its corresponding student and project center utilities: $\sum\limits_{a\in\mathcal{A}}\sum\limits_{i\in\mathcal{I}^a}\sum\limits_{p\in\mathcal{P}} \overline{w}_{aip}\overline{y}^2_{aip} + \sum\limits_{a\in\mathcal{A}}\sum\limits_{i\in\mathcal{I}^a}\sum\limits_{p\in\mathcal{P}} \underline{w}_{aip}\underline{y}^2_{aip} < \min\{1 + \gamma_1 q^+_{sp} + \gamma_2 k_{sp}\}$. Since $\min\{1 + \gamma_1 q^+_{sp} + \gamma_2 k_{sp}\} > 1$, penalty weights $\overline{w}_{aip}$ and $\underline{w}_{aip}$ that hold for $\sum\limits_{a\in\mathcal{A}}\sum\limits_{i\in\mathcal{I}^a}\sum\limits_{p\in\mathcal{P}} \overline{w}_{aip}\overline{y}^2_{aip} + \sum\limits_{a\in\mathcal{A}}\sum\limits_{i\in\mathcal{I}^a}\sum\limits_{p\in\mathcal{P}} \underline{w}_{aip}\underline{y}^2_{aip} < 1$, still retain the desired relationship. If there are $t_{aip}$ students with level $i$ in attribute $a$ assigned to project center $p$ already, the number of overassignment students deviated from the desire target cannot exceed $c_p - t_{aip}$. On the other hand, if there is no student with level $i$ in attribute $a$ assigned to project center $p$, the number of underassignment students deviated from the desire target in consideration is at most $t_{aip}$. Due to these upper bounds, we ensure that the maximum squared deviation penalty is strictly less than one: $\sum\limits_{a\in\mathcal{A}}\sum\limits_{i\in\mathcal{I}^a}\sum\limits_{p\in\mathcal{P}} \overline{w}_{aip}(c_p - t_{aip})^2 + \sum\limits_{a\in\mathcal{A}}\sum\limits_{i\in\mathcal{I}^a}\sum\limits_{p\in\mathcal{P}} \underline{w}_{aip}t^2_{aip} < 1$. The values of $\overline{w}_{aip}$ and $\underline{w}_{aip}$ can be

determined in the similar manner as in Appendix E.1.

For our experiments, we set all $\overline{w}_{aip}$ to zero and set all $\underline{w}_{aip}$ to be equal to one another, by constructing underassignment weights so that $\sum_{a\in\mathcal{A}}\sum_{i\in\mathcal{I}^a}\sum_{p\in\mathcal{P}}\underline{w}_{aip}t^2_{aip} < 1$ holds, setting $\underline{w}_{aip} = \frac{1}{\sum_{a\in\mathcal{A}}\sum_{i\in\mathcal{I}^a}\sum_{p\in\mathcal{P}}t^2_{aip}+\varepsilon}$, where $\varepsilon$ is a small positive value selected similar to the stable SPC formulation (4.1) of the WPI datasets $\forall\ a \in \mathcal{A},\ \forall\ i \in \mathcal{I}^a,\ \forall\ p \in \mathcal{P}$.

## E.3 Objective Coefficients for Synthetic Dataset in Section 4.5.5

We apply formulation (4.1) with the utility $u_{sp}$ set to the weighted student utility in which the value of $\gamma$ is chosen so that the model first prioritizes maximizing student placement, followed by maximizing student utility. We set the value of $\gamma$ to be the reciprocal of the minimum of the total capacity or the total number of students plus a small positive value, $\frac{1}{min\{\sum_{p\in\mathcal{P}}c_p,|\mathcal{S}|\}+\varepsilon}$; we set $\varepsilon$=1e-6. In this way, we ensure that $\gamma\sum_{s\in\mathcal{S}}\sum_{p\in\mathcal{P}}q_{sp} < 1$, that is, the contribution to the objective function of placing any single student outweighs the benefit of all student utility values combined.

# Appendix F

# Additional Experimental Results (Chapter 4)

Table 6.1 shows the result of stable SPC formulation (4.1) with different stability constraint sets across three years of the WPI datasets. Table 6.2 shows the result of the quadratic stable SPC formulation (4.2) with different stability constraint sets across three years of the WPI datasets.

| Year | 2017-2018 | | | | | 2018-2019 | | | | | 2019-2020 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Metric** | **Stability Constraint Type** | | | | | **Stability Constraint Type** | | | | | **Stability Constraint Type** | | | | |
| | **None** | **SPC1** | **SPC2** | **VVM** | **BBT** | **None** | **SPC1** | **SPC2** | **VVM** | **BBT** | **None** | **SPC1** | **SPC2** | **VVM** | **BBT** |
| Objective Value | **928.9766** | **927.9588** | 926.9637 | 927.9567 | 926.9637 | **927.9999** | **927.9999** | **927.9999** | **927.9999** | **927.9999** | 1,126.9002 | 1,106.8666 | **1,107.8621** | 1,107.8518 | 1,106.8571 |
| Best Bound | **928.9766** | **927.9615** | 928.9766 | 927.9761 | 928.9728 | **927.9999** | **927.9999** | **927.9999** | **927.9999** | **927.9999** | 1,126.9002 | 1,124.8810 | 1,126.8875 | **1,124.5777** | 1,126.8371 |
| MIPGap (%) | **0** | **0.0003** | 0.2172 | 0.0021 | 0.2167 | **0** | **0** | **0** | **0** | **0** | 0 | 1.6275 | 1.7173 | **1.5098** | 1.8051 |
| After Presolve | | | | | | | | | | | | | | | |
| #Constraints (Rows) | 966 | 13,835 | 20,145 | 15,564 | 10,979 | 974 | 12,007 | 15,818 | 12,130 | 7,641 | 1,183 | 12,583 | 16,304 | 13,630 | 8,355 |
| #Variables (Columns) | 14,359 | 18,280 | 24,230 | 15,307 | 12,397 | 11,169 | 15,407 | 18,199 | 12,137 | 8,970 | 12,597 | 16,560 | 18,298 | 13,796 | 10,371 |
| #Nonzeros | 28,710 | 158,553 | 95,178 | 1,896,966 | 2,026,000 | 22,338 | 107,719 | 74,355 | 1,314,575 | 1,080,987 | 25,194 | 84,965 | 77,151 | 1,441,600 | 1,360,148 |
| Model Density | 0.0021 | 0.0006 | **0.0002** | 0.0080 | 0.0149 | 0.0021 | 0.0006 | **0.0003** | 0.0089 | 0.0158 | 0.0017 | 0.0004 | **0.0003** | 0.0077 | 0.0157 |
| Total Students | 928 | 928 | 928 | 928 | 928 | 927 | 927 | 927 | 927 | 927 | 1,126 | 1,126 | 1,126 | 1,126 | 1,126 |
| Tier-1 Placements | 885 | **853** | **863** | 849 | **863** | **927** | **927** | **927** | **927** | **927** | 1,049 | **988** | **976** | 951 | 965 |
| Tier-2 Placements | 43 | **74** | 63 | 78 | 63 | 0 | 0 | 0 | 0 | 0 | 77 | 118 | **131** | 156 | 141 |
| Unassignments | 0 | **1** | 2 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 20 | **19** | **19** | 20 |
| Blocking Pairs | 150 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 293 | 0 | 0 | 0 | 0 |
| Time to Build Model (s) | 1.44 | 147.36 | 19.88 | 41.79 | 52.49 | 1.28 | 110.07 | **20.04** | 47.60 | 50.89 | 2.20 | 206.68 | 38.95 | 85.71 | 77.05 |
| First Incumbent Time (s) | 0.2 | 89 | 8,554 | 125 | 33,326 | 0.11 | 46.26 | **8.25** | 97.41 | 78.17 | 0.14 | 65 | 10,535 | 540 | 30,699 |
| Best Incumbent Time (s) | 0.2 | 75,264 | 28,319 | 5,763 | 43,313 | 0.11 | 46.26 | **8.25** | 97.41 | 78.17 | 0.14 | 86,400 | 85,305 | 31,707 | 86,400 |
| Run Time (s) | 0.2 | 86,400 | 86,400 | 86,400 | 86,400 | 0.11 | 46.26 | **8.25** | 97.41 | 78.17 | 0.14 | 86,400 | 86,400 | 86,400 | 86,400 |

Table 6.1: Comparison of SPC models with different stability constraint sets on three WPI datasets.

[4]Note that all runs timed out at 24 hours for the first and last years, while all runs solved well under the time limit for the second year, this is due to different flexibility of students in choosing project centers in each year. In particular, we observe that students are more restrictive in their selection of project centers in the first and last years; as the result, the process is more competitive and it poses more difficulty in solving the models.

| **Year** | 2017–2018 | | | | | | | | 2018–2019 | | | | | | | | 2019–2020 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Method** | SPC1 | | SPC2 | | VVM | | BBT | | SPC1 | | SPC2 | | VVM | | BBT | | SPC1 | | SPC2 | | VVM | | BBT | |
| **Consider Cohort** | **No** | **Yes** | **No** | **Yes** | **No** | **Yes** | **No** | **Yes** | **No** | **Yes** | **No** | **Yes** | **No** | **Yes** | **No** | **Yes** | **No** | **Yes** | **No** | **Yes** | **No** | **Yes** | **No** | **Yes** |
| Objective Value | **927.96** | 927.84 | **926.96** | 926.84 | **927.96** | 927.82 | **926.96** | 926.84 | **928.00** | 927.91 | **928.00** | 927.90 | **928.00** | 927.92 | **928.00** | 927.92 | **1,106.87** | 1,106.68 | **1,107.86** | 1,107.59 | **1,107.85** | 1,107.60 | **1,106.86** | 1,106.60 |
| Best Bound | **927.96** | 928.84 | **928.98** | 928.87 | **927.98** | 928.87 | **928.97** | 928.87 | **928.00** | 927.92 | **928.00** | 927.92 | **928.00** | 927.92 | **928.00** | 927.92 | **1,124.88** | 1,126.74 | **1,126.89** | 1,126.78 | **1,124.58** | 1,126.66 | **1,126.84** | 1,126.74 |
| MIPGap (%) | **0.0003** | 0.1076 | **0.2172** | 0.2198 | **0.0021** | 0.1133 | **0.2167** | 0.2198 | **0.0000** | 0.0006 | **0.0000** | 0.0023 | **0.0000** | 0.0005 | **0.0000** | 0.0005 | **1.6275** | 1.8124 | **1.7173** | 1.7327 | **1.5098** | 1.7203 | **1.8051** | 1.8200 |
| Presolved Model Size | 0.0006 | 0.0007 | 0.0002 | 0.0002 | 0.0080 | 0.0079 | 0.0149 | 0.0147 | 0.0006 | 0.0006 | 0.0003 | 0.0003 | 0.0089 | 0.0088 | 0.0158 | 0.0155 | 0.0004 | 0.0004 | 0.0003 | 0.0003 | 0.0077 | 0.0076 | 0.0157 | 0.0153 |
| Total Students | 928 | 928 | 928 | 928 | 928 | 928 | 928 | 928 | 927 | 927 | 927 | 927 | 927 | 927 | 927 | 927 | 1,126 | 1,126 | 1,126 | 1,126 | 1,126 | 1,126 | 1,126 | 1,126 |
| Tier-1 Placements | 853 | 851 | 863 | 860 | 849 | 850 | 863 | 863 | 927 | 920 | 927 | 927 | **927** | 921 | **927** | 919 | 988 | 979 | 976 | 975 | 951 | 946 | 965 | 965 |
| Tier-2 Placements | 74 | 76 | 63 | 66 | 78 | 77 | 63 | 63 | 0 | 7 | 0 | 0 | 0 | 6 | 0 | 8 | 118 | 127 | 131 | 132 | 156 | 161 | 141 | 141 |
| Unassignments | 1 | 1 | 2 | 2 | 1 | 1 | 2 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20 | 20 | 19 | 19 | 19 | 19 | 20 | 20 |
| Underassigned Females | 13 | **1** | 9 | **2** | 14 | **4** | 12 | **3** | 9 | **1** | 8 | **3** | 9 | **1** | 8 | **0** | 22 | **20** | 26 | **20** | 23 | **20** | 25 | **21** |
| Underassigned CS | 78 | **66** | 76 | **67** | 76 | **66** | 77 | **67** | 72 | **56** | 74 | **56** | 73 | **56** | 72 | **56** | 104 | **87** | 104 | **103** | 108 | **97** | 108 | **96** |
| Time to Build Model (s) | 147.36 | 129.85 | 19.88 | 18.48 | 41.79 | 46.47 | 52.49 | 49.98 | 110.07 | 98.26 | 20.04 | 16.62 | 47.60 | 45.76 | 50.89 | 45.63 | 206.68 | 234.86 | 38.95 | 37.29 | 85.71 | 78.17 | 77.05 | 92.78 |
| First Incumbent Time (s) | 89 | - | 8,554 | - | 125 | - | 33,326 | - | 46.26 | - | 8.25 | - | 97.41 | - | 78.17 | - | 65 | - | 10,535 | - | 540 | - | 30,699 | - |
| Best Incumbent Time (s) | 75,264 | 82,888 | 28,319 | 12,019 | 5,763 | 133 | 43,313 | 610 | 46.26 | 12,275 | 8.25 | 24,118 | 97.41 | 6,463 | 78.17 | 80,548 | 86,400 | 60,416 | 85,305 | 707 | 31,707 | 38,731 | 86,400 | 19,544 |
| Run Time (s) | 86,400 | 83,302 | 86,400 | 86,400 | 86,400 | 86,400 | 86,400 | 86,400 | 46.26 | 86,400 | 8.25 | 86,400 | 97.41 | 86,400 | 78.17 | 86,400 | 86,400 | 86,400 | 86,400 | 86,400 | 86,400 | 86,400 | 86,400 | 86,400 |

Table 6.2: Performance comparison of the Quadratic Stable SPC formulations with cohorts applying different stability constraint sets on three WPI datasets.