Inverse Integer Optimization With an Application in Recommender Systems

Mahsa Moghaddass

A Thesis
In the Department
of
Mechanical, Industrial, and Aerospace Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy (Industrial Engineering) at

Concordia University

Montreal, Qc, Canada

February 2020

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: Mahsa Moghaddass

Entitled: Inverse Integer Optimization With an Application in Recommender Systems

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Industrial Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Olga Ormandjieva

_____ External Examiner
Dr. Mustapha Nourelfath

_____ External to Program
Dr. Anjali Awasthi

_____ Examiner
Dr. MingYuan Chen

_____ Examiner
Dr. Hossein Hashemi Doulabi

_____ Supervisor
Dr. Daria Terekhov

Approved by _____
Dr. Ivan Contreras, Program Director
Department of Mechanical, Industrial and Aerospace Engineering

March 25, 2020

_____
Amir Asif, Dean
Gina Cody School of Engineering and Computer Science

ABSTRACT

Inverse Integer Optimization With an Application in Recommender Systems

Mahsa Moghaddass, Ph.D.

Concordia University, 2020

In typical (forward) optimization, the goal is to obtain optimal values for the decision variables given known values of optimization model parameters. However, in practice, it may be challenging to determine appropriate values for these parameters. Assuming the availability of historical observations that represent past decisions made by an optimizing agent, the goal of inverse optimization is to impute the unknown model parameters that would make these observations optimal (or approximately optimal) solutions to the forward optimization problem. Inverse optimization has many applications, including geology, healthcare, transportation, and production planning.

In this dissertation, we study inverse optimization with integer observation(s), focusing on the cost coefficients as the unknown parameters. Furthermore, we demonstrate an application of inverse optimization to recommender systems.

First, we address inverse optimization with a single imperfect integer observation. The aim is to identify the unknown cost vector so that it makes the given imperfect observation approximately optimal by minimizing the optimality error. We develop a cutting plane algorithm for this problem. Results show that the proposed cutting plane algorithm works well for small instances. To reduce the computational time, we propose an LP relaxation heuristic method. Furthermore, to obtain an optimal solution in a shorter amount of time, we combine both methods into a hybrid approach by initializing the cutting plane algorithm

with a solution from the heuristic method.

In the second study, we generalize the previous approach to inverse optimization with *multiple* imperfect integer observations that are all feasible solutions to one optimization problem. A cutting plane algorithm is proposed and then compared with an LP heuristic method. The results show the value of using multiple data points instead of a single observation.

Finally, we apply the proposed methods in the setting of recommender systems. By accessing past user preferences, through inverse optimization we identify the unknown model parameters that minimize an aggregate of the optimality errors over multiple points. Once the unknown parameters are imputed, the recommender system can recommend the best items to the users. The advantage of using inverse optimization is that when users are optimizing their decisions, there is no need to have access to a large amount of data for imputing recommender system model parameters. We demonstrate the accuracy of our approach on a real data set for a restaurant recommender system.

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere gratitude to my supervisor Dr. Daria Terekhov for the endless support of my Ph.D. study and related research for all her patience and motivation. She was not only my advisor but also as a loyal sympathizer that I had the chance to have her continuous support. I could not have achieved this research without her supports.

Besides thanking my advisor, I would like to thank the rest of my thesis committee: Dr. Mingyuan Chen, Dr. Hossein Hashemi, Dr. Ivan Contreras, Dr. Anjali Awasthi and Dr. Mustapha Nourelfath. I appreciate them not only for their time and insightful comments and encouragement but also for the hard questions which motivated me to widen my research from various perspectives.

I thank all my real friends and fellow lab-mates, especially for their advice and stimulating discussions and for all the fun we had in the last four years. I learned a lot from every single piece of advice and comment you shared with me.

Also, I would like to appreciate my family for all their supports and affections, who supported me every single day of my life in all aspects. Mainly, I would like to thank my brother Dr. Ramin Moghaddass who always helped me during these years that I was far from my parents. I always had his support behind me in different aspects. I will never forget his affections.

Last but not least, all my love, respect, and appreciations go to my parents, who sacrificed all their life for me and always supported me spiritually. Thank you for all your caring, concerns, and unconditional love. Whatever I have now is because of you, my dearest! All my heart dedicates this thesis to you! This way, maybe I can compensate for a few of your affections.

CONTRIBUTION OF AUTHORS

This dissertation presents a manuscript-based format. It contains three papers that have been submitted for publication or are close to submissions in the following journals. The first article titled "Inverse Integer Optimization with an Imperfect Observation" was submitted for publication in a journal in August 2019 and it is under the second review. The second manuscript, titled "Inverse Integer Optimization with Multiple Observations," was submitted for publication in a journal in March 2020. Finally, the third manuscript titled "Data-Driven Inverse Optimization for Recommender Systems," is also planned for submission to a journal. All three manuscripts are co-authored with Dr. Daria Terekhov, who established research guidelines and reviewed the papers before submission. For all of these works presented in this thesis, the author of this thesis acted as the principal researcher with the corresponding duties such as the developing mathematical models and algorithms, the programming of solution methods, and the analysis of computational results along with writing drafts of the papers.

DEDICATION

TO MY BELOVED PARENTS

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Inverse optimization (IO) is a study of learning the unknown parameter(s) of an optimization problem. Typically, in forward optimization, we assume that the values of the model parameters (i.e., objective function and constraint coefficients) are known, and the goal is to obtain optimal values for the decision variables given these parameters. In practice, it may be challenging to determine appropriate parameter values for the optimization model. Assuming the availability of historical observations that represent past decisions made by an optimizing agent, the goal of IO is to impute the values of the unknown model parameters in order to make these observations optimal solutions to the forward optimization model.

For instance, in a production planning problem in which the goal is to find out the total level of production that minimizes the overall cost, the back-order cost may be unknown and hard to estimate. However, past observations, such as the number of items produced in the past few years, are typically available. Consider the situation that these past production quantities were set by a knowledgeable manager who has an intuition for the back-order costs; we want to automate future decision-making while mimicking the manager's intuition as well as the importance of back-order costs in their decision-making process. In this

case, we can use past production levels to determine the back-order cost using inverse optimization [Troutt et al., 2006]. Once the unknown parameters are estimated, a forward optimization model with the imputed parameters will determine optimal levels of production that reflect the manager's goals and intuition. This idea of using IO can be generalized to many other applications, such as healthcare [Chan et al., 2014, Boutilier et al., 2015], economics[Keshavarz et al., 2011, Bertsimas et al., 2015, Saez-Gallego and Morales, 2017], energy [Turner and Chan, 2013], and transportation [Bertsimas et al., 2015].

There are a few publications in the literature focusing on employing IO as a technique for identifying the constraint coefficients or the right-hand-side parameters of the optimization problem (e.g., Chan and Kaw [2019], Troutt et al. [2005], Saez-Gallego and Morales [2017]). However, most papers in the literature aim to impute the unknown parameters of the objective function (i.e., the cost vector) in a linear or a convex model (e.g., Chan et al. [2014], Aswani et al. [2018], Wang [2009], Schaefer [2009]). In this thesis, the goal of IO is learning the cost vector in an integer problem with imperfect observation(s).

In the rest of this chapter, we discuss different types of IO problems classified first based on the nature of the observations and then based on various applications. Section 1.2 provides the motivations of this thesis. Section 1.3 gives an overview of the methods that we propose for inverse integer optimization with imperfect data. We describe the organization of this dissertation in Section 1.4. We present a summary of this dissertation's main contributions in Section 1.5.

1.1 A LITERATURE REVIEW BASED ON CLASSIFICATIONS OF INVERSE OPTIMIZATION

We start by describing the literature through three classifications based on the nature of the given observations. First, the classical definition of IO focuses on the problems where

the given observation is a candidate optimal. In contrast, a modern interpretation of IO focuses on the existence of noise in the observation. The second classification focuses on the inclusion of a single observation versus multiple observations. The third classification considers the integrality of the observations. Finally, we can also classify the IO literature based on various applications.

1.1.1 Noise-Free Versus Noisy Inverse Optimization

There can be different assumptions regarding the optimality of the observation. In a classical definition of the IO problem, some authors, such as Ahuja and Orlin [2001], Erkin et al. [2010], Wang [2009] and Schaefer [2009], assume that the given observation is a candidate optimal solution and thus we refer to it as *noise-free*. In linear programming, this means that the observed solution is a point on the boundary of the feasible region, and we are guaranteed to obtain a cost vector that makes this solution optimal. Based on the notation used in Ahuja and Orlin [2001], let $\mathbf{c}$ be as a specified cost vector and $\mathbf{x}^0$ a given feasible solution that may or may not be an optimal solution of an optimization problem $\mathbf{P}$. The goal is to perturb the cost vector $\mathbf{c}$ to $\mathbf{d}$ such that $\mathbf{x}^0$ becomes an optimal solution concerning the perturbed cost vector $\mathbf{d}$ and the cost of perturbation $\|\mathbf{d}-\mathbf{c}\|_p$ is minimum for a chosen L_p norm. However, the existence of noise in data is unavoidable in practice, and real data are imperfect [Chan et al., 2019, Aswani et al., 2018]. In the context of IO, noise in the data may stem from the data collection process or represent that decision-makers deviate from acting optimally. Also, considering noisy data can be very important when there are multiple observations: it may not be possible to make all of them simultaneously optimal for the same cost vector.

Since the classical inverse linear optimization framework cannot accommodate for data points that are in the interior of the feasible region or are infeasible [Goli, 2015, Chan

et al., 2019], the studies moved into the modern approach to overcome this limitation. A modern definition of IO allows the given observation to be *noisy*, where the goal is to estimate model parameters to make the given solution *approximately* optimal [Chan et al., 2019, Troutt et al., 2006, Esfahani et al., 2018, Aswani et al., 2018]. For example, in a linear optimization problem, *noisy* can mean that the given feasible solution is strictly in the interior of the feasible region.

1.1.2 Inverse Optimization for A Single Observation Versus Multiple Observations

In this subsection, we review available studies in the literature based on the number of observations used for the IO problem. While there are papers that consider a single observation [Ahuja and Orlin, 2001, Wang, 2009, Schaefer, 2009], recent work is more focused on including multiple observations as prior data [Esfahani et al., 2018, Aswani et al., 2018, Keshavarz et al., 2011, Babier et al., 2019]. The reason for such a trend is that multiple observations contain more information regarding the feasible region of the problem. In general, having more prior data for IO results in obtaining better parameter estimates.

1.1.3 Continuous Observation Versus Integer Observation

Unlike continuous IO, where the assumption is that the observations are continuous [Chan et al., 2019, Aswani et al., 2018, Bertsimas et al., 2015, Esfahani et al., 2018, Keshavarz et al., 2011], integer IO refers to those problems where the forward problem is an integer optimization problem, i.e., the decision variables are integer.

For inverse integer optimization, the researchers proposed both exact and heuristic algorithms to solve the corresponding discrete optimization problems. Wang [2009] presented a cutting plane algorithm for solving the inverse mixed-integer linear programming problem,

which is to minimally perturb the objective function of a mixed-integer linear program to make a given feasible solution exactly optimal. Schaefer [2009] also considered the integer programming version of IO. Using super-additive duality, they provided a polyhedral description of the set of inverse feasible objectives. Bulut and Ralphs [2016] focused on evaluating the complexity of inverse mixed-integer linear optimization. A paper by Duan [2009] proposed a heuristic method in parallel to the cutting plane algorithm to overcome the limitations of cutting plane methods.

In Table 1.1, we summarize the main references in the IO literature based on the above classifications. This table displays that several papers have considered a *candidate optimal* integer observation for IO. However, to the best of our knowledge, there is no published IO work that considers *error* for an *integer* observation. We address the case when there is *integer error* in the observed integer solution, i.e., the observation is a feasible solution to the forward integer problem. Since the term "noisy" generally encompasses non-integer error, we refer to our case as inverse integer optimization with *"imperfect"* data.

References	Noise-free	Noisy	Single	Multiple	Integer	Continuous
Ahuja and Orlin [2001]	X		X			X
Erkin et al. [2010]	X		X			X
Troutt et al. [2006]		X	X			X
Troutt et al. [2005]		X	X			X
Chan et al. [2019]		X	X			X
Aswani et al. [2018]		X		X		X
Bertsimas et al. [2015]		X		X		X
Saez-Gallego and Morales [2017]		X		X		X
Babier et al. [2019]		X		X		X
Esfahani et al. [2018]		X		X		X
Keshavarz et al. [2011]		X		X		X
Goli [2015]		X	X			X
Schaefer [2009]	X		X		X	
Wang [2009]	X		X		X	X
Bulut and Ralphs [2016]	X		X		X	X
Duan [2009]	X		X		X	X
This Dissertation		**X**		**X**	**X**	

Table 1.1: Categorization of the Literature Based on the Nature of Observations

1.1.4 Applications of Inverse Optimization

IO has applications in a variety of settings. *Geophysical scientists* were among the earliest researchers that investigated IO, and utilized it in predicting the movements of earthquakes [Tarantola, 1987]. Since then, IO has been employed in a wide spectrum of applications including but not limited to *healthcare, econometrics, energy, transportation, network design and control, machine scheduling, and production planning*. For example, in the *healthcare* setting, IO has been employed for Intensity-Modulated Radiation Therapy (IMRT) [Chan and Kaw, 2019, Chan et al., 2014, Lee et al., 2013, Boutilier et al., 2015]. IMRT is a technique for delivering radiation to a tumor. Using historical treatment plans (i.e., dose distributions) developed by radiologists/physicians, the IO method can identify the optimal concealed weights of the objective function of the IMRT model.

Many IO references have demonstrated applications in *network* optimization and control. One of the applications that Bertsimas et al. [2015] presented was an application of IO in *transportation science*. Given a particular road network, one typically specifies a cost function and then calculates the resulting flow under user equilibrium. In their paper, the goal of IO was to estimate the congestion function on a road network, given the observed data equilibria to make the observed data an approximately optimal solution to the optimization problem. Their estimate helps either to predict congestion on the network in the future or to inform subsequent network design problems. In the work of Ahuja and Orlin [1998], the objective was to impose the minimum total toll to make the user equilibrium flow identical to the system optimal flow. Finding the *utility function in economics* based on past customer purchasing behavior is another application of IO [Keshavarz et al., 2011]. Saez-Gallego and Morales [2017] used IO for finding the utility functions of price-responsive *energy* consumers. Turner and Chan [2013] applied IO as a method for improving the design of Leadership in *Energy* and Environmental Design (LEED) rating

system. IO also has applications in *machine scheduling*. For example, in Li et al. [2013], typical values of the due dates are given, and they are to be modified to achieve a target value of maximum lateness. Troutt et al. [2006] and Troutt et al. [2008] employed IO to discover cost coefficients for production planning problems since, in practice, organizations often have difficulty in precisely articulating their goals and the technological coefficients of their constraints [Troutt et al., 2008]. Table 1.2 summarizes some of the papers in IO based on their applications. To the best of our knowledge, there is no previous work that focuses on the application of IO in *recommender systems*, which is the main application focus of this thesis.

Reference	Application
Tarantola [1987]	Geology (predicting movements of the earthquakes)
Chan and Kaw [2019] Goli [2015] Boutilier et al. [2015] Chan et al. [2014] Lee et al. [2013]	Healthcare
Bertsimas et al. [2015] Chow and Recker [2012] Faragó et al. [2003] Ahuja and Orlin [1998] Zhang and Liu [1996]	Transportation & Network Optimization
Saez-Gallego and Morales [2017] Bertsimas et al. [2015] Keshavarz et al. [2011]	Economics
Turner and Chan [2013]	Energy
Li et al. [2013] Zhang et al. [2011]	Scheduling
Troutt et al. [2008] Troutt et al. [2006] Troutt [1995]	Production Planning
This Dissertation	**Recommender Systems**

Table 1.2: Categorizations of the Literature, Based on Applications

1.2 MOTIVATIONS

In this section, we discuss the motivations of this thesis, given the reviewed literature. The thesis develops and evaluates new methods for solving IO problems with *imperfect integer* observation(s) to find the unknown cost coefficients of optimization problems.

1.2.1 Inverse Optimization with a *Single Imperfect Integer* Observation

Chan et al. [2019] presented an IO model where one seeks to obtain the cost coefficients of an optimization problem to make the given solution optimal. If the optimality conditions are satisfied, then there is no error in the fit. Chan et al. [2019] suggests a definition for cases when the optimality conditions cannot be satisfied. In this case, there is a positive error, and the observation for the linear optimization problem is an interior point of the feasible region instead of being on the boundary.

Wang [2009] studied inverse integer optimization when there is an integer observation that is a candidate optimal for the original problem.

Thus, IO for continuous noisy observation and integer noise-free observation are discussed by Chan et al. [2019] and Wang [2009], respectively. To the best of our knowledge, there is no work in the literature on IO problems with *imperfect integer observation*. This thesis develops IO models for such cases.

1.2.2 Inverse Optimization with *Multiple Imperfect Integer* Observations

Incorporating more data can potentially help find more precise estimates of the unknown parameters of optimization models. Hence, in addition to considering a single imperfect integer observation, a generalization to multiple integer observations is studied as well.

In this case, the existence of the error is unavoidable. This error stems from two different scenarios. The first scenario is that not all of the multiple observations lie on the boundary of the feasible region of the forward integer problem. Consequently, the goal is to find a cost vector that makes these observations approximately optimal. Another scenario is that all of these multiple observations are on the boundary of the feasible region of the forward integer problem but not on the same hyperplane. In both cases, finding a cost vector that makes all observations simultaneously optimal is impossible. Therefore, we let the observations in the model to be *imperfect* and will minimize the aggregate error. To the best of knowledge, there is no prior published work on IO with *multiple imperfect integer* observations.

1.2.3 Inverse Optimization for Recommender Systems

Although the results proposed in this thesis are general and practical for different applications, the application focus of the thesis is on IO in recommender systems (RS). To the best of our knowledge, there is no prior work that uses IO in RS. RS is a technique that recommends the best items to users based on past user preferences. There are many data-driven techniques for RS. The proposed work is unique as it requires less amount of data compared to machine learning techniques. Unlike many machine learning models used for recommended systems, our approach focuses on optimal decision making.

1.3 PROPOSED METHODS

1.3.1 Inverse Optimization with Imperfect Integer Observation(s)

First, we propose a mathematical model for IO with one imperfect integer observation. Given an integer observation, the goal is to learn the unknown coefficients of the objective function of a forward optimization problem to make the given observation optimal or

approximately optimal. In the IO model, the objective is to minimize the optimality error. We define the optimality error as the difference between the objective values of the "given solution" and the "optimal solution" for a particular cost vector. The optimality error also will further referred to as the "optimality gap".

Subsequently, we generalize the IO model to multiple integer observations while allowing observations to be imperfect. Given past data, the goal is to obtain the unknown cost vector of the forward problem such that make the observations optimal or close to optimal.

Later, to apply the model for RS, we consider multiple feasible regions (one for each user) and develop an IO model for multiple users. The goal is to find a matrix of the importance of the attributes to minimize the aggregate of the optimality gaps. Using this matrix, we can recommend the best items to users in the future. The only inputs are past users' preferences (ratings). Also, the assumption is that the attributes of the users and items are available.

1.3.2 Cutting Plane and Heuristic Algorithms

We develop a cutting plane algorithm that can find a cost vector that makes the given integer solution approximately optimal by minimizing the optimality gap(s) between the given solution(s) and an optimal integer solution.

We propose an LP relaxation heuristic method to reduce the computational time. We compare the results from the cutting plane algorithm with the heuristic approach.

To leverage the speed of the heuristic and the quality of solutions found, we also develop and evaluate a hybrid method that initializes the cutting plane method with the heuristic solution.

1.3.3 Applying Inverse Optimization in Recommender Systems

In RS, the user/item attribute weight matrix has a significant role. Since this matrix or any prior knowledge about this matrix is often unknown, we use an IO framework to estimate it given user and item attributes and also past users preferences. After obtaining this matrix from an inverse model, it can be used in the forward problem to recommend the best items to any user given their attributes. Using an IO approach, not only can we find the essential attributes and the relationships between them that affect users' decisions, but also we can recommend the best items to the users. For this purpose, we use data from multiple users. This application is a particular case of IO with multiple integer observations where the observations are binary.

1.4 ORGANIZATION OF THIS DISSERTATION

Chapter 1 gives a brief introduction to IO. In this chapter, we discuss the definition of IO and its applications. Also, we explain different aspects of IO based on several categorizations. We then present the motivations for this work and a summary of the proposed solution methods. We will end up this chapter by a summary of contributions made in the thesis.

Based on the identified gaps in the literature, Chapter 2 starts by defining an IO problem with a single integer observation that can be imperfect. We develop an IO model for a single imperfect observation to obtain the unknown coefficients of the objective function to minimize the optimality gap. We propose a cutting plane algorithm and a heuristic method to initialize the algorithm to reduce CPU time. After analyzing the results, we discuss different scenarios that can happen based on the location of the observations. We illustrate why the heuristic can potentially discover a reasonable result close to optimal.

Chapter 3 focuses on multiple integer observations. In this chapter, we demonstrate

why having multiple observations is important and can provide a better result for the problem. Since for multiple observations, it is not always possible to simultaneously make them all optimal; we let the observations be imperfect. We then propose a cutting plane algorithm and heuristic methods for multiple imperfect integer observations. In the implementation part, we show how the quality of the solution will change with an increasing number of observations and also compare the results with the time there is just a single given observation.

Since in Chapters 2 and 3 we considered theoretical and implementation aspects of IO for imperfect integer observation(s), in Chapter 4 we demonstrate an application of these methods in recommender systems. In this problem, user and item attributes are available. The user/item attributes weights matrix is unknown. Instead, there are some past users' ratings for the items. By learning from these ratings and the user and item attributes, IO can identify this unknown matrix. Learning this matrix helps to recommend the best items to a future user. We use a publicly available dataset to apply the proposed framework to investigate whether we can recommend the best restaurants to the users. We divide the dataset into a training set and a test set, and after obtaining the matrix from the training set, we apply it in the test set to determine how close the predicted results are to the actual data in the test set.

We conclude this dissertation in Chapter 5 by summarizing the contributions and describing future work ideas. Figure 1.1 illustrates a summary of the chapters of this thesis.

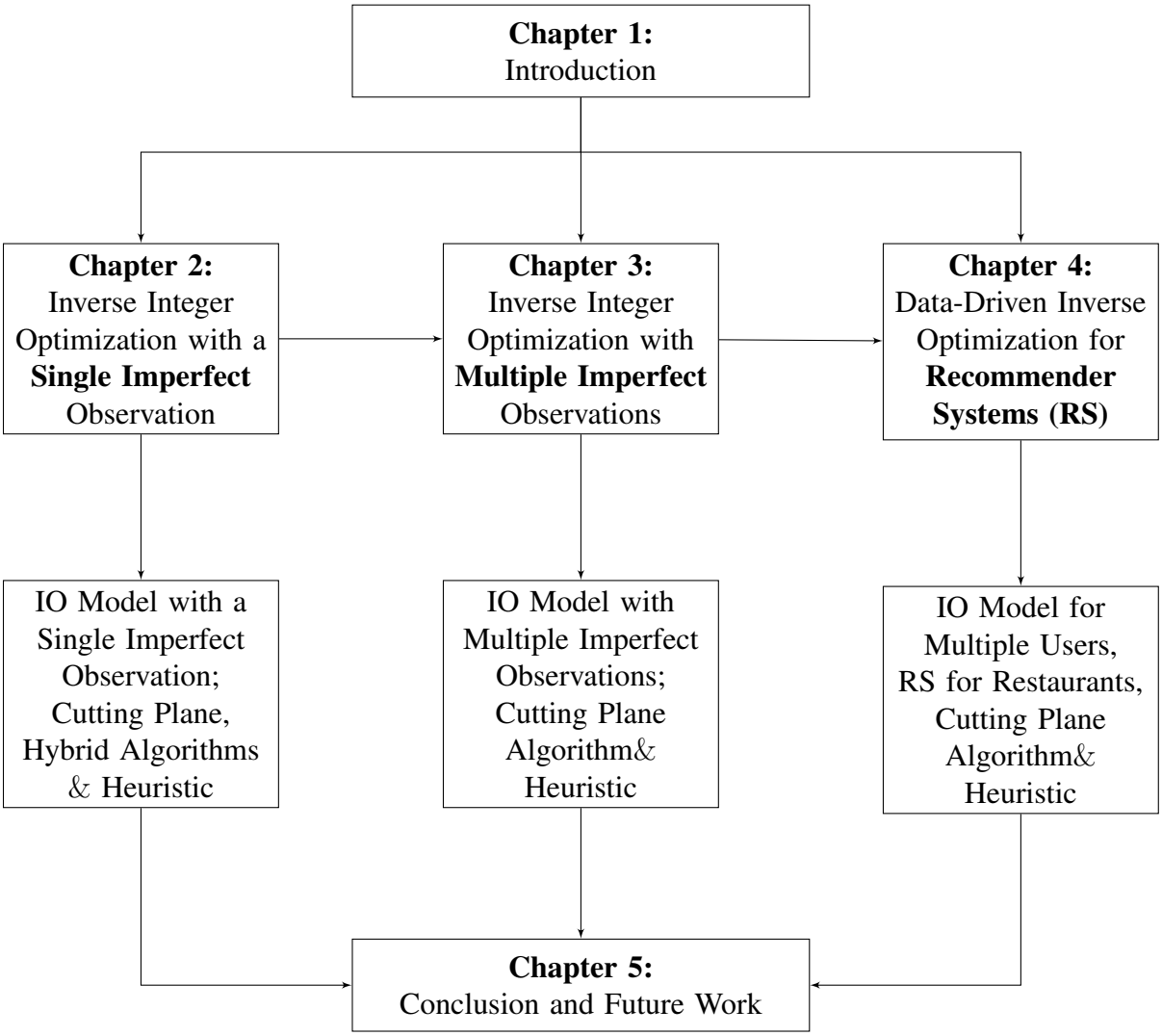


Figure 1.1: Organization of the Thesis

1.5 SUMMARY OF CONTRIBUTIONS

The contributions of this dissertation can be summarized as follows:

1. IO with an imperfect integer observation

    - Developing an IO model for a single imperfect integer observation.
    - Proposing a cutting plane algorithm for solving the IO model.

- Presenting an LP relaxation heuristic method for reducing computational time.
- Introducing a hybrid method with initializing the cutting plane algorithm by a heuristic method.
- Comparing the results obtained from cutting plane algorithm, heuristic and hybrid methods based on CPU time and optimality gaps.
- Analysing different solutions of IO based on the location of the observations.

2. IO with multiple imperfect integer observations

    - Generalizing the previous IO model for multiple integer observations.
    - Proposing a cutting plane algorithm for solving an IO model with multiple integer observations.
    - Recommending an LP relaxation heuristic method for reducing computational time.
    - Comparing the results obtained from cutting plane algorithm with the heuristic based on CPU time and optimality gaps.
    - Comparing the results obtained from a single observation and multiple observations to check the changes in the optimality gaps.
    - Analysing different solutions of IO based on the location of the observations.

3. Data-driven IO with an application in recommender systems

    - Developing a customized IO model for RS.
    - Proposing a cutting plane algorithm for solving IO in RS using multiple observations.
    - Proposing a heuristic method for IO in RS.

- Demonstrating the possibility of using a small amount of data for recommending when users are optimizing their decisions.
- Employing a real data set for recommending restaurants and showing the validation of using IO as a technique for recommending items to the users by obtaining high accuracy percentages of predictions.

Chapter 2

Inverse Integer Optimization with an Imperfect Observation

2.1 ABSTRACT

In contrast to a typical optimization problem, the main objective of inverse optimization is to find unknown parameters of a forward problem from an (approximately) optimal observed solution to that problem. This study develops and evaluates methods for solving inverse *integer* optimization problems with an *imperfect* observation for learning the *cost coefficients*. We propose a cutting plane algorithm for this problem and compare it to a heuristic method, i.e., solving the inverse of the linear relaxation of the forward problem. We then propose a hybrid approach that initializes the cutting plane algorithm from the solution of the inverse of the relaxation. To the best of our knowledge, this is the first study to evaluate a general approach for the inverse integer problem with an imperfect observation.

2.2 INTRODUCTION

Given a past observation of decision variable values as input, inverse optimization (IO) aims to impute parameters of an optimization model that generated the observation [Ahuja and Orlin, 2001]. A recent trend in IO is solving problems with imperfect/noisy data. In this case, the observations are not a candidate optimal to the postulated forward model [Aswani et al., 2018, Babier et al., 2019, Esfahani et al., 2018, Chan et al., 2019], which can be due to a variety of reasons, including a decision-maker's bounded rationality or a noise-prone measurement process. However, the majority of recent papers study continuous inverse optimization problems.

There are a few papers in the literature with the focus of studying inverse integer optimization (IIO) [Wang, 2009, Schaefer, 2009, Bulut and Ralphs, 2016, Duan, 2009, Turner and Chan, 2013]. Importantly, all of these papers assume that a) a prior estimate of the unknown cost vector is given, and the objective is to find the smallest perturbation of that cost vector to make the observed solution optimal, and b) the given (integer) observation lies on the boundary of the integral hull (which in turn implies the existence of a cost vector that would make this point optimal). We relax both of these assumptions in the current paper. Following four out of the five published IIO papers, namely Wang [2009], Schaefer [2009], Bulut and Ralphs [2016], Duan [2009], we employ a cutting plane approach.

This study proposes an IO problem in which all elements of a given observation are integer but subject to error and thus *imperfect*. In this work, an *imperfect* observation is a feasible integer solution to the forward problem that may not be a candidate for being an optimal solution. This observation is a feasible integer solution that is *not* necessarily on the boundary of the integral hull, or is precluded from being optimal by constraints imposed on the cost vector in the inverse problem. We develop a cutting plane algorithm that can find a cost vector that makes the given integer solution minimally sub-optimal by minimizing

the absolute optimality gap between the given solution and an optimal integer solution. We compare our approach to a heuristic approach, i.e., solving the inverse of the linear programming relaxation, whose performance, to the best of our knowledge, also has not been evaluated in the literature. To leverage the speed of the heuristic and the surprisingly good solutions it finds, we also develop and evaluate a hybrid method which initializes the cutting plane method with the heuristic solution.

2.3 METHODOLOGY

2.3.1 Forward Optimization Model

Given $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, and $\mathbf{A} \in \mathbb{R}^{m \times n}$, the *forward* integer optimization problem is

$$\mathbf{FIO}(\mathbf{c}) : \underset{\mathbf{x}}{\text{minimize}} \quad \mathbf{c}'\mathbf{x} \tag{1a}$$

$$\text{subject to} \quad \mathbf{A}\mathbf{x} \geq \mathbf{b} \tag{1b}$$

$$\mathbf{x} \in \mathbb{Z}^n. \tag{1c}$$

Note that we use $'$ to denote transpose throughout this thesis; for example, in (1a) above, $\mathbf{c}'$ is the transpose of $\mathbf{c}$. For convenience, we define $\mathcal{X} = \{\mathbf{x} \mid \mathbf{A}\mathbf{x} \geq \mathbf{b}, \mathbf{x} \in \mathbb{Z}^n\}$ and $conv(\mathcal{X})$ as the convex hull of $\mathcal{X}$, also referred to as the integral hull. We assume $conv(\mathcal{X}) \neq \emptyset$ and is bounded. We let $\mathcal{X}^{\text{OPT}}(\mathbf{c})$ be the set of optimal solutions to $\mathbf{FIO}(\mathbf{c})$, $\mathcal{X}^{\text{OPT}} := \cup_{\mathbf{c} \neq \mathbf{0}} \mathcal{X}^{\text{OPT}}(\mathbf{c})$ and $\mathbb{S} \subseteq \mathcal{X}^{\text{OPT}}$ be the set of all (integer) extreme points of $conv(\mathcal{X})$.

2.3.2 Inverse Optimization Model

Given an $\hat{\mathbf{x}} \in \mathcal{X}$, our goal is to find a $\mathbf{c}$ vector that makes $\hat{\mathbf{x}}$ a minimally sub-optimal solution to $\mathbf{FIO}(\mathbf{c})$. The model we propose combines the absolute duality gap formulation of Chan et al. [2019], which finds a $\mathbf{c}$ vector that minimizes the absolute duality gap for a

continuous linear program (LP) given a feasible $\hat{\mathbf{x}}$, and the IIO formulation of Wang [2009], which assumes $\hat{\mathbf{x}} \in \mathcal{X}^{\text{OPT}}$ and also assumes knowledge of a prior cost vector. Our model aims to minimize an analog of the absolute duality gap for the integer case, assuming that $\hat{\mathbf{x}}$ may not be in $\mathcal{X}^{\text{OPT}}$. The resulting "imperfect" inverse integer optimization (IIIO) problem is stated as follows:

$$\mathbf{IIIO}_0(\hat{\mathbf{x}}): \quad \underset{\mathbf{c}}{\text{minimize}} \quad \mathbf{c}'\hat{\mathbf{x}} - \mathbf{c}'\mathbf{x}^*(\mathbf{c}) \tag{2a}$$

$$\text{subject to} \quad \mathbf{x}^*(\mathbf{c}) \in \arg\min_{\mathbf{x}}\{\mathbf{c}'\mathbf{x} \mid \mathbf{x} \in \mathcal{X}\} \tag{2b}$$

$$\|\mathbf{c}\|_1 = 1. \tag{2c}$$

This bilevel model finds a $\mathbf{c}$ vector that minimizes the optimality gap between $\hat{\mathbf{x}}$ and $\mathbf{x}^*(\mathbf{c})$, i.e., it finds $\mathbf{c}$ that makes $\hat{\mathbf{x}}$ minimally sub-optimal to $\mathbf{FIO}(\mathbf{c})$. We reformulate this model in terms of $\mathbb{S}$:

$$\mathbf{IIIO}_1(\hat{\mathbf{x}}): \quad \min_{\mathbf{c}} \ \max_{\mathbf{x}^s \in \mathbb{S}} \ (\mathbf{c}'\hat{\mathbf{x}} - \mathbf{c}'\mathbf{x}^s) \tag{3a}$$

$$\text{subject to} \quad \|\mathbf{c}\|_1 = 1. \tag{3b}$$

$\mathbf{IIIO}_1(\hat{\mathbf{x}})$ minimizes, over all $\mathbf{c}$, the maximum of the difference between the objective values of the points in $\mathbb{S}$ and $\hat{\mathbf{x}}$. If $\hat{\mathbf{x}} \in \mathcal{X}^{\text{OPT}}$, then $\max_{\{\mathbf{x}^s \in \mathbb{S}\}}(\mathbf{c}'\hat{\mathbf{x}} - \mathbf{c}'\mathbf{x}^s) = 0$ since $\mathbf{c}'\hat{\mathbf{x}} \leq \mathbf{c}'\mathbf{x}^s$ for all $\mathbf{x}^s \in \mathbb{S}$. In an earlier paper, Wang [2009] proposed a model for minimally adjusting a given cost vector to a new cost vector that would make an observed point exactly optimal. Given $\hat{\mathbf{x}} \in \mathcal{X}^{\text{OPT}}$, their model relies on the constraint $\mathbf{c}'\hat{\mathbf{x}} \leq \mathbf{c}'\mathbf{x}^s, \forall \mathbf{x}^s \in \mathbb{S}$, to ensure optimality of $\mathbf{c}$ for the inverse problem. We note that if $\hat{\mathbf{x}} \notin \mathcal{X}^{\text{OPT}}$, then their model is infeasible and their cutting plane algorithm does not converge: in particular, there is no vector $\mathbf{c}$ such that $\mathbf{c}'\hat{\mathbf{x}} \leq \mathbf{c}'\mathbf{x}^s, \forall \mathbf{x}^s \in \mathbb{S}$, since for any candidate $\hat{\mathbf{c}}$ there exists at least one

$\mathbf{x}^s$, i.e., the one optimal for $\hat{\mathbf{c}}$, such that $\hat{\mathbf{c}}'\hat{\mathbf{x}} > \hat{\mathbf{c}}'\mathbf{x}^s$.

In principle, $\mathbf{IIIO}_1(\hat{\mathbf{x}})$ will find a $\mathbf{c}$ that makes $\hat{\mathbf{x}}$ exactly optimal if $\hat{\mathbf{x}} \in \mathcal{X}^{\text{OPT}}$, and will otherwise find a $\mathbf{c}$ that minimizes the *optimality error*, i.e., the error between an optimal integer solution and the given $\hat{\mathbf{x}}$. However, $\mathbf{IIIO}_1(\hat{\mathbf{x}})$ cannot be directly used to solve the problem as it requires full knowledge of $\mathbb{S}$. In Section 2.3.3, we, therefore, propose a cutting plane algorithm that can overcome this issue. Another weakness of $\mathbf{IIIO}_1(\hat{\mathbf{x}})$ is that it has no explicit characterization of the $\mathbf{c}$ vector other than the normalization constraint, which in preliminary experiments has led to an inefficient cutting plane approach. To remedy this problem, we adopt Wang's idea of keeping the dual feasibility constraints in the integer IO model [Wang, 2009], just as they are used in linear inverse optimization. We also linearize the objective, resulting in the following model:

$$\mathbf{IIIO}_2(\hat{\mathbf{x}}): \quad \underset{\mathbf{c},\mathbf{y}}{\text{minimize}} \quad z \tag{4a}$$

$$\text{subject to} \quad z \geq \mathbf{c}'\hat{\mathbf{x}} - \mathbf{c}'\mathbf{x}^s \qquad \forall \mathbf{x}^s \in \mathbb{S} \tag{4b}$$

$$\mathbf{A}'\mathbf{y} = \mathbf{c}, \mathbf{y} \geq \mathbf{0} \tag{4c}$$

$$\|\mathbf{c}\|_1 = 1, \tag{4d}$$

where (4c) are the dual feasibility constraints from the dual of the LP relaxation of $\mathbf{FIO}(\mathbf{c})$.

Theorem 1 *Given* $\hat{\mathbf{x}} \in \mathcal{X}$, $\mathbf{c}$ *is optimal to* $\mathbf{IIIO}_0(\hat{\mathbf{x}})$ *if and only if* $(\mathbf{c}, \mathbf{y})$ *is optimal to* $\mathbf{IIIO}_2(\hat{\mathbf{x}})$.

Proof. First, $\mathbf{c}$ is optimal to $\mathbf{IIIO}_0(\hat{\mathbf{x}})$ if and only if $\mathbf{c}$ is optimal to $\mathbf{IIIO}_1(\hat{\mathbf{x}})$ since $\mathbf{x}^*(\mathbf{c}) \in \mathcal{X}^{\text{OPT}}(\mathbf{c}) \iff \mathbf{c}'\mathbf{x}^s \geq \mathbf{c}'\mathbf{x}^*(\mathbf{c})\ \forall \mathbf{x}^s \in \mathbb{S} \iff \mathbf{c}'\hat{\mathbf{x}} - \mathbf{c}'\mathbf{x}^s \leq \mathbf{c}'\hat{\mathbf{x}} - \mathbf{c}'\mathbf{x}^*(\mathbf{c})\ \forall \mathbf{x}^s \in \mathbb{S} \iff \max_{\mathbf{x}^s \in \mathbb{S}}\{\mathbf{c}'\hat{\mathbf{x}} - \mathbf{c}'\mathbf{x}^s\} \leq \mathbf{c}'\hat{\mathbf{x}} - \mathbf{c}'\mathbf{x}^*(\mathbf{c})$. Expressions (4a) and (4b) in $\mathbf{IIIO}_2(\hat{\mathbf{x}})$ constitute a standard reformulation of the objective function of $\mathbf{IIIO}_1(\hat{\mathbf{x}})$. Therefore, $\mathbf{c}$ is optimal to $\mathbf{IIIO}_0(\hat{\mathbf{x}})$ if and only if $\mathbf{c}$ is optimal to $\mathbf{IIIO}_2(\hat{\mathbf{x}})$ without constraint (4c). It

remains to show that there exists a $\mathbf{y}$ satisfying (4c). For any $\mathbf{c}$, if $\mathbf{x}^*(\mathbf{c}) \in \mathcal{X}^{\text{OPT}}(\mathbf{c})$ then $\mathbf{x}^*(\mathbf{c}) \in \{\mathbf{x} \mid \mathbf{A}\mathbf{x} \geq \mathbf{b}\}$. Since this LP relaxation is feasible and bounded, the dual of the LP relaxation is feasible and bounded, and there exists $\mathbf{y}$ such that $\mathbf{A}'\mathbf{y} = \mathbf{c}, \mathbf{y} \geq \mathbf{0}$.

2.3.3 Cutting Plane Algorithm

To solve $\mathbf{IIIO}_2(\hat{\mathbf{x}})$ without enumerating all extreme points, we propose a cutting plane algorithm, shown as Algorithm 1. This method solves two optimization problems in each iteration. The master problem is an IO model: a version of $\mathbf{IIIO}_2(\hat{\mathbf{x}})$ with $\mathbb{S}$ consisting of the extreme points that have been considered by the algorithm up to the current iteration. For ease of exposition, we assume, without loss of generality, that the extreme points of $\mathbb{S}$ are numbered according to the order that the cutting plane algorithm finds them. Thus, after obtaining a cost vector $\mathbf{c}^t$ in current iteration t from solving model (4) with a restricted set of extreme points, a separation problem, i.e., model (1), is solved with the current $\mathbf{c}^t$ in the same iteration to find an optimal solution $\mathbf{x}^t$ to $\mathbf{FIO}(\mathbf{c}^t)$.

There are two conditions for termination, depending on whether $\hat{\mathbf{x}}$ is imperfect or not. If the inequality $\mathbf{c}'^t\hat{\mathbf{x}} \leq \mathbf{c}'^t\mathbf{x}^t$ is satisfied, the algorithm terminates and the given observation $\hat{\mathbf{x}}$ is an optimal solution of model (1). Otherwise, by adding the extreme point $\mathbf{x}^t$ to the set $\mathbb{S}$, a valid inequality constraint (4b) is added to the inverse model (4), and after updating the current best error and the current best cost vector, the inverse and forward models will be resolved in next iteration. If $\hat{\mathbf{x}}$ is imperfect, the algorithm terminates whenever the error $\mathbf{c}'^t\hat{\mathbf{x}} - \mathbf{c}'^t\mathbf{x}^t$ at iteration t is equal to the best error found up to and including that iteration and the obtained $\mathbf{c}^t$ is the same as the one that generated the best error. Note that we show a pseudo-code for Algorithm 1 and some implementation details are omitted; for instance, lines 11 and 12 require comparison of floating points numbers and thus are implemented not using a simple equality but rather using tolerance of 0.01 for errors in line 11 and

Algorithm 1 A Cutting Plane Algorithm for IO with an Imperfect Integer Observation

$t \leftarrow 0, \mathbb{S}' \leftarrow \emptyset$ ▷ Initialization
$\mathbf{c}^0 \leftarrow \mathbf{IIIO}_2(\hat{\mathbf{x}}, \mathbb{S}')$
$\mathbf{x}^0 \leftarrow \mathbf{FIO}(\mathbf{c}^0)$
$\epsilon^{\text{best}} \leftarrow \mathbf{c}'^0\hat{\mathbf{x}} - \mathbf{c}'^0\mathbf{x}^0$
$\mathbf{c}^{\text{best}} \leftarrow \mathbf{c}^0$
while $\mathbf{c}'^t\hat{\mathbf{x}} > \mathbf{c}'^t\mathbf{x}^t$ **do** ▷ Optimality condition
    $\mathbb{S}' \leftarrow \mathbb{S}' \cup \mathbf{x}^t$
    $t \leftarrow t + 1$
    $\mathbf{c}^t \leftarrow \mathbf{IIIO}_2(\hat{\mathbf{x}}, \mathbb{S}')$ ▷ Master problem
    $\mathbf{x}^t \leftarrow \mathbf{FIO}(\mathbf{c}^t)$ ▷ Separation problem
    if $\mathbf{c}'^t\hat{\mathbf{x}} - \mathbf{c}'^t\mathbf{x}^t = \epsilon^{\text{best}}$ **then**
        if $\mathbf{c}^t = \mathbf{c}^{\text{best}}$ **then**
            Stop
        else
            $\mathbf{c}^{\text{best}} \leftarrow \mathbf{c}^t$
        end if
    end if
    if $\mathbf{c}'^t\hat{\mathbf{x}} - \mathbf{c}'^t\mathbf{x}^t < \epsilon^{\text{best}}$ **then**
        $\epsilon^{\text{best}} \leftarrow \mathbf{c}'^t\hat{\mathbf{x}} - \mathbf{c}'^t\mathbf{x}^t$
        $\mathbf{c}^{\text{best}} \leftarrow \mathbf{c}^t$
    end if
end while
Output: $\epsilon^* \leftarrow \epsilon^{\text{best}}$ $\mathbf{c}^* \leftarrow \mathbf{c}^{\text{best}}$

0.000001 for cost vector floating point values.

This algorithm is inspired by the cutting plane algorithm of Wang [2009] which was developed for the case when the given solution is a candidate optimal and a prior cost vector estimate is known. Since Wang's algorithm assumes the given solution is a candidate optimal, it will not find any feasible solution until the last iteration (i.e., as soon as a feasible solution is found, it is optimal as well); however, our algorithm will find a feasible solution in each iteration and will stop when a $\mathbf{c}$ is found that minimizes the optimality gap between the observed solution and an optimal integer solution, i.e., $\mathbf{c}'\hat{\mathbf{x}} - \mathbf{c}'\mathbf{x}^*(\mathbf{c})$, where $\mathbf{x}^*(\mathbf{c})$ is as defined in Equation (2b).

Theorem 2 *Algorithm 1 terminates finitely with an optimal solution to* $\mathbf{IIIO}_2(\hat{\mathbf{x}})$.

Let $\mathbf{IIIO}_2(\mathbb{S}', \hat{\mathbf{x}})$ be a special case of $\mathbf{IIIO}_2(\hat{\mathbf{x}})$ where $\mathbb{S}$ in Equation (4b) is replaced by $\mathbb{S}' \subseteq \mathbb{S}$. The following lemma helps to prove convergence of Algorithm 1 and also shows that to find a cost vector that makes $\hat{\mathbf{x}}$ minimally sub-optimal to $\mathbf{FIO}(\mathbf{c})$, it may be sufficient to consider only a subset of extreme points $\mathbb{S}'$.

Lemma 1 *Let* $(\mathbf{c}^s, \mathbf{x}^s)$ *be a solution obtained by Algorithm 1 in any iteration* $s = 1, ..., f$, *where* $\mathbf{c}^f$ *is an optimal solution to* $\mathbf{IIIO}_2(\mathbb{S}', \hat{\mathbf{x}})$ *and* $\mathbf{x}^f \in \mathcal{X}^{OPT}(\mathbf{c}^f)$. *If* $\mathbf{x}^f \in \mathbb{S}'$, $\mathbf{c}'^f\hat{\mathbf{x}} - \mathbf{c}'^f\mathbf{x}^f = \epsilon^{best}$ *and and* $\mathbf{c}^f = \mathbf{c}^{best}$, *then* $\mathbf{c}^f$ *is global optimal for* $\mathbf{IIIO}_2(\mathbb{S}, \hat{\mathbf{x}})$.

Proof. We need to show that $\mathbf{x}^f$ is an extreme point that achieves the optimal value of z in (4a) given $\mathbf{c}^f$. To derive a contradiction, assume that it does *not* do so, which implies that there exists $\bar{\mathbf{x}}$ and $\bar{\mathbf{c}}$ such that $\bar{z} = \bar{\mathbf{c}}'\hat{\mathbf{x}} - \bar{\mathbf{c}}'\bar{\mathbf{x}}$ and $\bar{z} < \epsilon^{best}$. Two cases are possible: either $\bar{\mathbf{x}}$ and $\bar{\mathbf{c}}$ were discovered before iteration f or they would be discovered after iteration f if the algorithm did not terminate. In the first case, if $\bar{\mathbf{x}}$ and $\bar{\mathbf{c}}$ were found in iteration $s < f$ then ϵ^{best} would have been updated before iteration f, contradicting $\mathbf{c}^{f\prime}\hat{\mathbf{x}} - \mathbf{c}^{f\prime}\mathbf{x}^f = \epsilon^{\text{best}}$. In the second case, by optimality of $\bar{\mathbf{x}}$ and $\bar{\mathbf{c}}$, $\bar{z} = \bar{\mathbf{c}}'\hat{\mathbf{x}} - \bar{\mathbf{c}}'\bar{\mathbf{x}} \geq \bar{\mathbf{c}}'\hat{\mathbf{x}} - \bar{\mathbf{c}}'\mathbf{x}^s \ \forall s \in \mathbb{S}$ and $\bar{z} = \bar{\mathbf{c}}'\hat{\mathbf{x}} - \bar{\mathbf{c}}'\bar{\mathbf{x}} < \mathbf{c}^{f\prime}\hat{\mathbf{x}} - \mathbf{c}^{f\prime}\mathbf{x}^f$, which is a contradiction since $\mathbf{x}^f \in \mathbb{S}'$.

Proof of Theorem 2. At every iteration s a point $\mathbf{x}^s \in \mathbb{S}$ is found by solving $\mathbf{FIO}(\mathbf{c}^s)$. Given the current $\mathbb{S}'$, if $\mathbf{x}^s \in \mathbb{S}'$, $\mathbf{c}'^s\hat{\mathbf{x}} - \mathbf{c}'^s\mathbf{x}^s = \epsilon^{\text{best}}$ and $\mathbf{c}^f = \mathbf{c}^{best}$, then by Lemma 1, an optimal solution to $\mathbf{IIIO}_2(\hat{\mathbf{x}})$ has been found. Otherwise, none of the termination criteria of the algorithm will be met, and so $\mathbf{x}^s$ will be added to $\mathbb{S}'$. Since the number of integer points in $\mathbb{S}$ is finite, the algorithm will find an optimal solution in a finite number of iterations.

2.3.4 Heuristic

$\mathbf{IIIO}_0(\hat{\mathbf{x}})$ can also be solved heuristically by solving the IO formulation for the linear programming relaxation of $\mathbf{FIO}(\mathbf{c})$. In this paper, we refer to this approach as the *LP heuristic*; the specific IO formulation we use is the absolute duality gap formulation of Chan et al. [2019]. After obtaining a cost vector $\mathbf{c}^H$ from that formulation, we solve the forward integer optimization (model 1) to find the corresponding optimal integer solution $\mathbf{x}^H(\mathbf{c}^H)$.

2.3.5 Hybrid Heuristic-Cutting Plane Algorithm

We also propose a *hybrid* approach, which initializes the cutting plane algorithm with the solution from the heuristic. This algorithm starts by solving the absolute duality gap formulation of Chan et al. [2019] instead of $\mathbf{IIIO}_2(\hat{\mathbf{x}})$ to find an initial cost vector $\mathbf{c}^0$. It then initializes the best error and the best cost vector and adds the obtained extreme point $\mathbf{x}^s$ to the set $\mathbb{S}'$ and re-solves $\mathbf{IIIO}_2(\mathbb{S}', \hat{\mathbf{x}})$. The rest of the algorithm and the optimality condition are the same as Algorithm 1.

2.4 EXPERIMENTAL RESULTS

In this section, we present the results of three sets of experiments. The first one uses our exact and heuristic methods to solve problems where the error is due to sub-optimality of $\hat{\mathbf{x}}$; the second one compares the result of the cutting plane algorithm (Algorithm 1) with the hybrid method. Finally, the third one considers error due to misspecification of the prior objective function. All methods are implemented in a Visual C++ 2013 environment using Concert Technology to call IBM ILOG CPLEX version 12.7.0.0. on a PC with 3.50 GHz and 16.0 GB of RAM. Problem instances of six different sizes are created by sampling points from $\mathcal{N}(10, 30)$ and computing their convex hull via the *scipy.spatial.convexhull*

package [QHull Library, 2018]. We note that most instances generated had a dense $\mathbf{A}$ matrix. For each size, we considered 20 different instances, and each specific instance was solved for ten different imperfect observations. All observations are feasible integer interior points sampled randomly from the subset of integer points close to the boundary (i.e., close to optimal).

To implement constraint (4d) in CPLEX, we use the function IloAbs via IloCplex in a C++ application. Despite the constraint being non-linear, IloAbs is one of the constraints that can be converted to a mixed-integer formulation and handled by CPLEX. For reference, please see https://www.ibm.com/support/knowledgecenter/SSSA5P_12.8.0/ilog.odms.cplex.help/CPLEX/UsrMan/topics/discr_optim/logical_constr/08_expr.html.

2.4.1 Experiment 1: Comparison of Cutting Plane Algorithm and LP-Relaxation-Based Heuristic

For Experiment 1, we have solved a set of randomly generated instances using both our proposed cutting plane algorithm and the LP relaxation-based heuristic. The results presented in Table 2.1 compare computational time (rounded to the nearest second), number of iterations, and optimality gap. From Table 2.1, we observe that the method requires under 10 seconds of computational time on average for problems with up to 60 constraints and 5 variables; computational time increases substantially for problems of size 90 by 10. We also attempted to solve instances with more than 10 variables – for these instances, even an initial feasible solution could not be found within the time limit of 12 hours. Such behavior is due to the difficulty of the forward problem – CPLEX could not find a feasible solution to the initial forward problem within the given time limit. While we leave complexity analysis of the problem for future work, we conjecture that it is NP-complete, following Theorem 8 of Bulut and Ralphs [2016] which shows the NP-completeness of

the inverse mixed-integer LP lower-bounding problem for the case with a prior $\mathbf{c}$ and a candidate-optimal observation.

Since there is no prior published work proposing an exact method for an IIO problem with an imperfect observation, our experiment provides an initial understanding of how efficiently the problem can be solved if optimality is important. Moreover, this experiment evaluates the performance of the LP relaxation heuristic – although solving the inverse of the LP relaxation is a natural approach in the absence of an exact algorithm, prior published work has not evaluated how well this method performs on general IIO problems. In particular, Table 2.1 shows that the optimality gap between the heuristic and exact solution is very small. In terms of CPU time, the LP heuristic is substantially faster – not surprising given that the cutting plane method needs more time as it has to solve two optimization problems for every iteration while the heuristic solves only one optimization problem in total; the run-time of the cutting plane algorithm increases with the number of iterations. We further discuss this observation in Section 2.5. Similar to previous results for the *candidate optimal* version of the problem [Wang, 2009, Schaefer, 2009], we conclude that the cutting plane algorithm for IIO is limited to solving small instances. However, our results also suggest that the heuristic is a good option, especially in large scale problems if a guarantee of optimality is not required.

2.4.2 Experiment 2: Comparison of Cutting Plane Algorithm and the Hybrid Approach

Given the fast run-time and good-quality solutions of the heuristic seen in Table 2.1, next, we evaluate a hybrid method that finds a good solution quickly (due to heuristic) but still has the opportunity to prove optimality (due to the cutting plane algorithm). As shown in Figures 2.1 and 2.2, the hybrid algorithm is faster than the original cutting plane

Size	Cutting Plane Algorithm			LP Heuristic		
M x N	Avg. # of Iters	Avg. CPU Time (Seconds)	Avg. Optimality Gap $\mathbf{c'^{*}}\hat{\mathbf{x}} - \mathbf{c'^{*}}\mathbf{x}^{*Int}$	Avg. CPU Time (Seconds)	Avg. Optimality Gap $\mathbf{c'^{H}}\hat{\mathbf{x}} - \mathbf{c'^{H}}\mathbf{x}^{*Int}$	Avg. (%) Relative Error
4 x 2	6.27	2	0.73	< 1	0.89	20.90
5 x 2	6.70	2	0.84	< 1	0.98	19.20
20 x 5	27.26	9	0.52	< 1	0.53	1.71
40 x 5	28.84	9	0.52	<1	0.54	3.29
60 x 5	29.97	9	0.52	< 1	0.54	4.30
90 x 10	67.70	280	0.37	< 1	0.37	0.11

Table 2.1: Cutting Plane and LP Heuristic Performance on Randomly Generated Instances

algorithm in terms of average CPU time and the number of iterations, respectively. Furthermore, the hybrid method found and proved an optimal solution for all problem instances for which the cutting plane was able to do so, as shown in Table 2.2. Therefore, due to the decrease in computational time, it is worth to use the hybrid algorithm rather than the original cutting plane approach. The heuristic, in contrast, while remaining the fastest method with good quality solutions, could not find an optimal solution in 95% of our instances, as shown in Table 2.2.

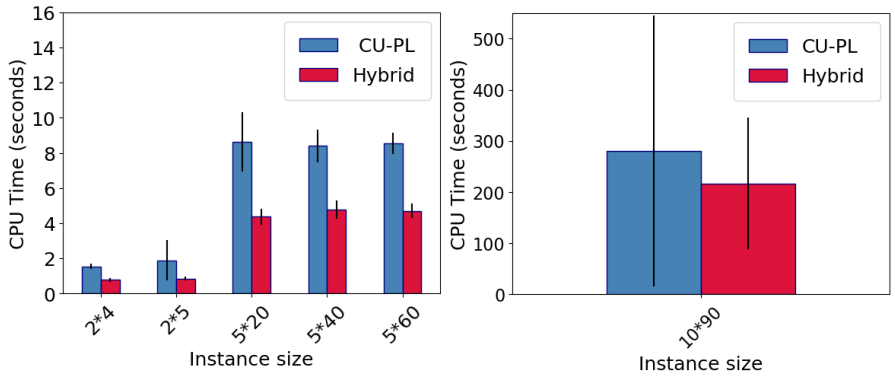


Figure 2.1: Average CPU Time for Cutting Plane Algorithm and the Hybrid

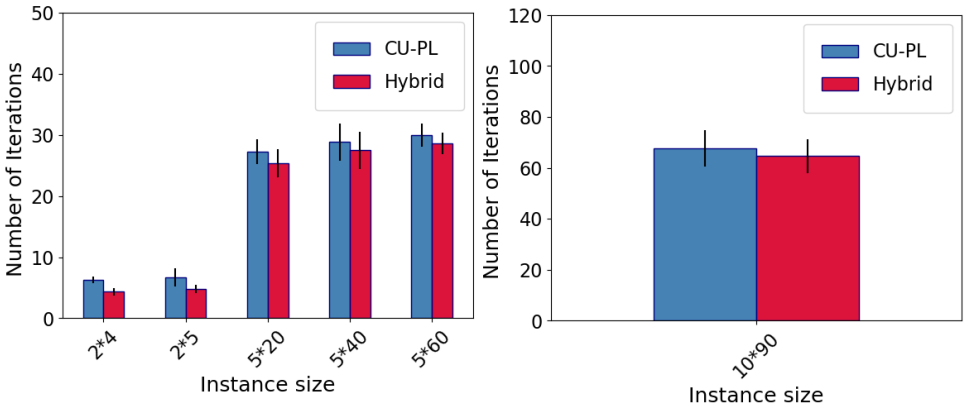


Figure 2.2: Average Number of Iterations for Cutting Plane Algorithm and the Hybrid

2.4.3 Experiment 3: Error Due to Misspecification of Prior Objective Function

In Experiment 3, we added constraints on the cost vector for a large-scale problem in order to show the use of the model for the case when the error is not coming from the given observation, but rather from constraints on the cost vector. In particular, we used the integer problem *gt2* from MIPLIP [miplib2017]. All variables are integer, some of these integer variables are binary. There are 405 constraints with 188 variables. We solved the inverse of this problem 20 times, each time with a randomly generated integer solution on the boundary of the feasible region. In each case, we included some constraints on $\mathbf{c}$ such as

Instance Size \ Optimal Found	LP-Heur	Cu-Pl	Hybrid
• **M= 4 N= 2**	2	200	200
• **M= 5 N= 2**	9	200	200
• **M= 20 N= 5**	5	200	200
• **M= 40 N= 5**	6	200	200
• **M= 60 N= 5**	2	200	200
• **M= 90 N= 10**	31	200	200
• **Total**	55	1200	1200

Table 2.2: Number of Optimal Solutions Found by the Heur, Cu-Pl & Hybrid (/1200)

enforcing some elements of $\mathbf{c}$ to be strictly greater than 0. These results showed an average (over the 20 IO instances) optimality gap of 0.0015, which is caused by the constraints on $\mathbf{c}$ despite the points being on the boundary.

2.5 DISCUSSION

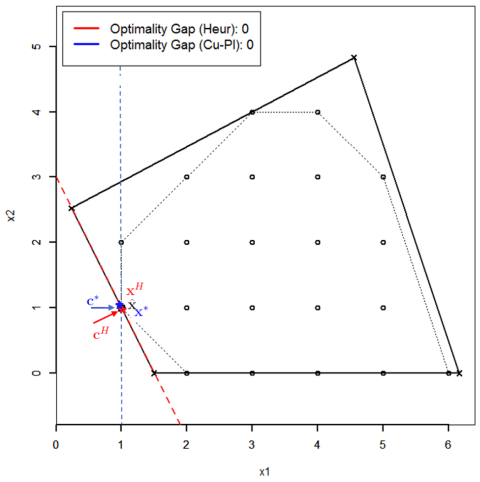
Given that in (forward) integer programming, the LP relaxation does not work in general well for solving the integer problem, one may expect the same to hold for the *inverse* integer problem. However, our experimental results have shown that the solution obtained from the inverse LP relaxation performs well for the integer problem. Although a full characterization requires further investigation, here we provide some examples demonstrating both the cases when the inverse LP relaxation finds an optimal solution to the inverse IP

and the cases where it does not.

From the perspective of the LP, the given integer point is simply a feasible, perhaps imperfect solution, i.e., the inverse LP treats $\hat{\mathbf{x}}$ the same as any other feasible solution. By applying the results of Chan et al. [2019], we know that there exists an optimal solution to the inverse LP found by projecting $\hat{\mathbf{x}}$ to the closest boundary of the feasible region, where "closest" in our case is measured in terms of absolute optimality gap. Since obtaining $\mathbf{c}^H$ is governed by the optimality gap with respect to the LP, $\mathbf{FIO}(\mathbf{c}^H)$ will find the integer solution $\mathbf{x}^H$ that is closest to the boundary of the LP; however, $\mathbf{x}^H$ may not be the candidate optimal integer solution closest to $\hat{\mathbf{x}}$.

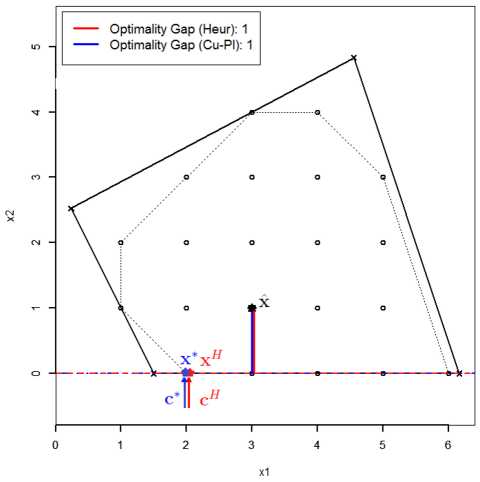
2.5.1 Scenarios Where Heuristic Returns an Optimal Solution

In some cases, the optimality gaps obtained from cutting plane algorithm and LP heuristic are the same, i.e., the heuristic finds an optimal solution. Figure 2.3 shows a candidate optimal $\hat{\mathbf{x}}$ on the boundary of both LP and IP: although the cost vectors obtained from the two methods are different, both of them make $\hat{\mathbf{x}}$ optimal. In Figure 2.4, $\hat{\mathbf{x}}$ is an interior point with $\mathbf{c}^* = \mathbf{c}^H$, which implies that the resulting optimality gaps are the same. These results are not surprising, since in the former scenario, $\hat{\mathbf{x}}$ is a candidate optimal solution to both the LP and the IP, while in the latter scenario, $\hat{\mathbf{x}}$ is projected onto the boundary of the LP relaxation which includes integer solutions.



$\hat{\mathbf{x}}$=(1,1), $\mathbf{c}^*$=(1,0), $\mathbf{c}^H$=(0.67, 0.33)

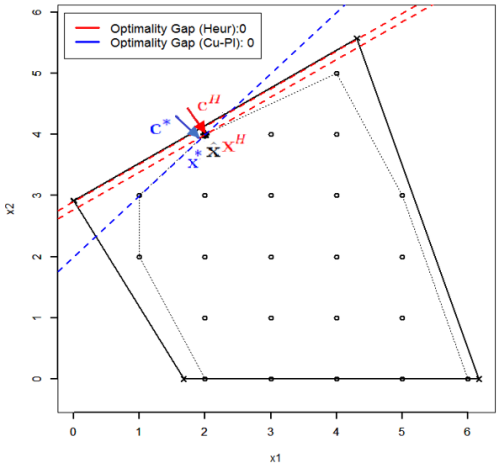
Figure 2.3: Point on the IP & LP Boundary



$\hat{\mathbf{x}}$=(3,1), $\mathbf{c}^*$=(0, 1), $\mathbf{c}^H$=(0, 1)

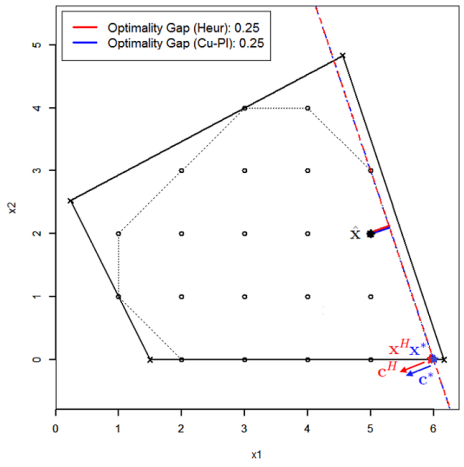
Figure 2.4: Interior Point

In Figure 2.5, $\hat{\mathbf{x}}$ is on the boundary of IP but not LP. Although the cost vectors obtained from LP and IP are not the same, both make $\hat{\mathbf{x}}$ optimal, since $\hat{\mathbf{x}}$ is the integer solution that minimizes the absolute LP duality gap. In Figure 2.6 $\hat{\mathbf{x}}$ is an interior point but the closest LP boundary and the hyperplane through the two nearest integer extreme points are parallel, resulting in $\mathbf{c}^* = \mathbf{c}^H$ and equal (but non-zero) optimality gaps.



$\hat{\mathbf{x}}$=(2,4), $\mathbf{c}^*$=(0.5, -0.5), $\mathbf{c}^H$=(0.38, -0.619)

Figure 2.5: Point on the Boundary of IP

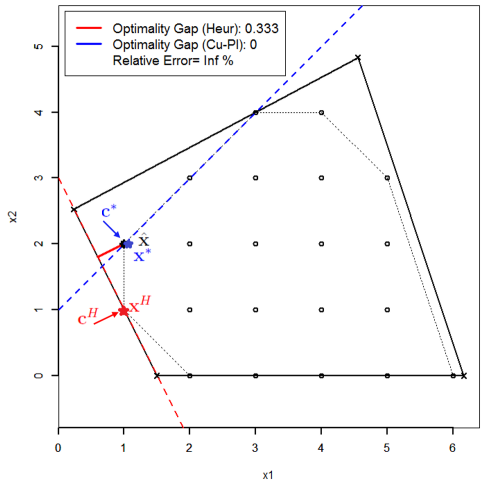


$\hat{\mathbf{x}}$=(5,2), $\mathbf{c}^*$=(-0.75, -0.25), $\mathbf{c}^H$=(-0.75, -0.25)

Figure 2.6: Interior Point Close to IP Bound

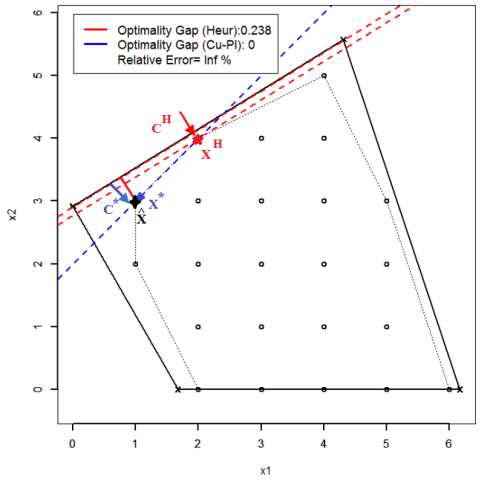
2.5.2 Scenarios Where Heuristic Does Not Return an Optimal Solution

Figures 2.7-2.10 illustrate cases in which the heuristic is unable to find an optimal cost vector. In all these cases, we see that $\mathbf{x}^H$ and $\mathbf{x}^*$ lie on the same hyperplane but that the cost vectors are not the same, resulting in $\mathbf{c}^H$ being sub-optimal for the integer inverse problem. Except for Figure 2.10, $\mathbf{x}^H$ is always not the closest integer solution to $\hat{\mathbf{x}}$. Figure 2.10, however, shows that even when $\mathbf{x}^H$ coincides with $\mathbf{x}^*$, the cost vector returned by the heuristic can be sub-optimal. This case also illustrates why line 12 of Algorithm 1 is necessary and why in Lemma 1 it is not enough to find an extreme point that is already in $\mathbb{S}$: since our objective is optimality gap, both the correct $\mathbf{c}$ and $\mathbf{x}(\mathbf{c})$ need to be found.



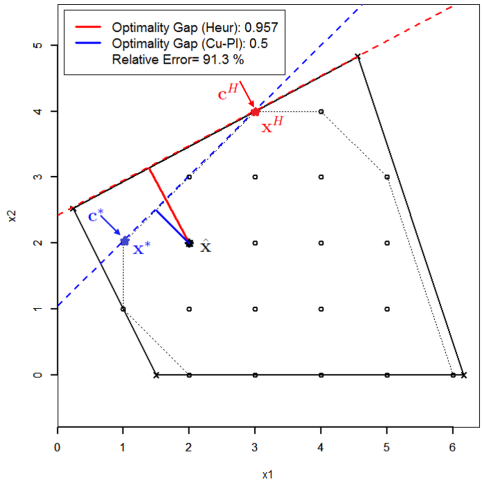
$\hat{\mathbf{x}}$=(1,2), $\mathbf{c}^*$=(0.5,-0.5), $\mathbf{c}^H$=(0.67, 0.33)

Figure 2.7: Extreme Point of IP Case 1



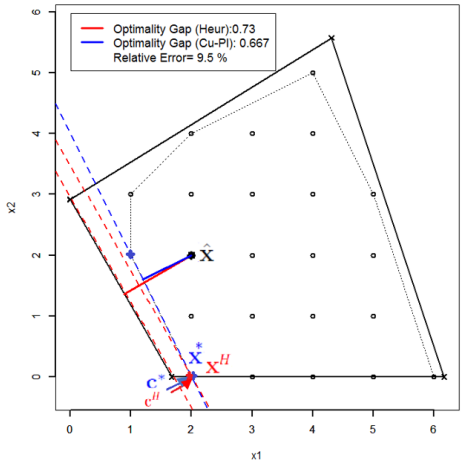
$\hat{\mathbf{x}}$=(1,3), $\mathbf{c}^*$=(0.5,-0.5), $\mathbf{c}^H$=(0.38, -0.62)

Figure 2.8: Extreme Point of IP Case 2



$\hat{\mathbf{x}}$=(2,2), $\mathbf{c}^*$=(0.5,-0.5), $\mathbf{c}^H$=(0.34, -0.65)

Figure 2.9: Interior Point Case 1

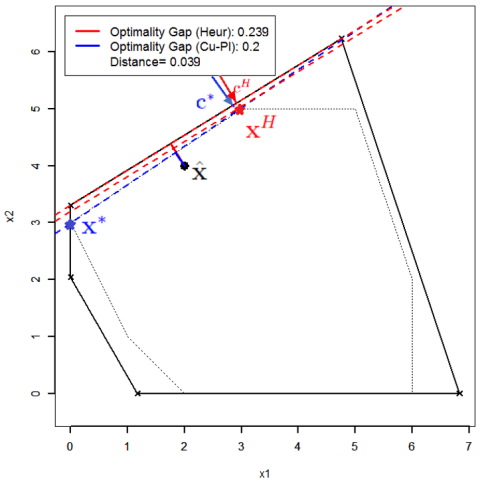


$\hat{\mathbf{x}}$=(2,2), $\mathbf{c}^*$=(0.67, 0.33), $\mathbf{c}^H$=(0.63, 0.36)

Figure 2.10: Interior Point Case 2

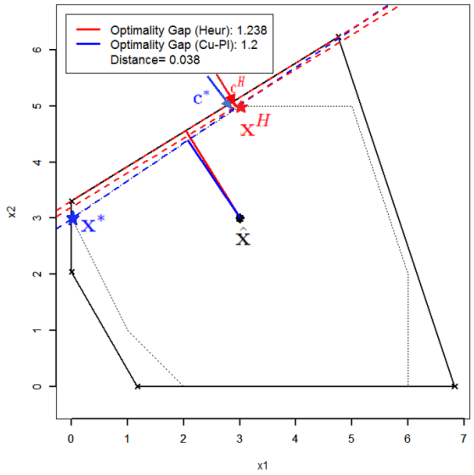
2.5.3 Interior Point Location

Our initial conjecture was that the closer an integer point is to the centroid of the feasible region, the worse the heuristic would perform. However, it is easy to construct a counter-example to this conjecture, as we do in the following two figures. Figure 2.12 demonstrates a deep integer point which is close to the core of the integer convex hull, while Figure 2.11 shows an integer point close to the boundary. Although in Figure 2.12 the optimality gaps from both methods are bigger, Figure 2.11 shows that LP heuristic still works well regardless of the location of the point; the difference in optimality gaps is a bit smaller in Figure 2.12 (i.e., heuristic is a bit better in this case).



$\hat{\mathbf{x}}$=(2,4), $\mathbf{c}^*$=(0.4,-0.6), $\mathbf{c}^H$=(0.380, -0.619)

Figure 2.11: Point Close to the Boundary



$\hat{\mathbf{x}}$=(3,3), $\mathbf{c}^*$=(0.4, -0.6), $\mathbf{c}^H$=(0.380, -0.619)

Figure 2.12: Point Close to the Centre

2.6 CONCLUSION

This paper proposes three methods for solving an IO problem with an imperfect integer observation: a cutting plane algorithm, an LP-relaxation-based heuristic, and a hybrid which initializes the cutting plane method with the heuristic solution. The inverse problem with an imperfect integer observation has not been previously addressed in the literature. Furthermore, to the best of our knowledge, there have been no previously published results evaluating the performance of the inverse LP-relaxation approach for the inverse integer problem. While the cutting plane algorithm is an exact method, our results show that the heuristic works better in terms of computational time and finds good quality solutions on our experimental test set. The hybrid method takes advantage of both: it finds a close-to-optimal solution quickly due to the heuristic and is still an exact method due to the structure of the cutting plane algorithm. Future work may include developing ways to avoid adding repetitive cuts to reduce the CPU time for larger instances. Besides, with real data, a single

data point may not be representative and thus extending the model and algorithm to multiple observations, as in the continuous inverse literature [Aswani et al., 2018, Babier et al., 2019, Esfahani et al., 2018], is important.

Chapter 3

Inverse Integer Optimization with Multiple Observations

3.1 ABSTRACT

Inverse optimization is the technique of learning the unknown model parameters of an optimization problem using past observed solutions to that problem. These given integer observations can be imperfect. Hence, the objective of inverse optimization is to minimize the aggregate of optimally errors to make the observations optimal or close to optimal. An exact method using a cutting plane algorithm is proposed and then compared with an LP heuristic method. The results show the value of using multiple observations instead of a single one.

3.2 INTRODUCTION

In a forward linear optimization model, cost coefficients $\mathbf{c}$, right-hand side vector $\mathbf{b}$, and the constraint matrix $\mathbf{A}$ are known parameters. Given these parameters, we solve the forward problem to find an optimal solution. However, in practice, some or all of the model

parameters (e.g., $\mathbf{c}$, $\mathbf{A}$, $\mathbf{b}$) may not be easily measurable, or we might not be sure about the accuracy of our estimates. Instead, past observed solutions of the forward problem may be available. Given these observations, the goal of an inverse optimization problem is to impute the unknown model parameters of the forward problem to make the given observations exactly or approximately optimal.

The inverse optimization (IO) literature makes a distinction between the case when observations are assumed to be candidate optimal solutions for the forward problem [Ahuja and Orlin, 2001, Wang, 2009, Schaefer, 2009] and the case when they might be noisy [Troutt et al., 2006, Esfahani et al., 2018, Aswani et al., 2018, Chan et al., 2019]. For instance, in a linear optimization problem, a given observation is a candidate optimal solution if it lies on the boundary of the feasible region; similarly, in an integer linear program, an observation is candidate optimal if it is an integer solution on the boundary of the integral hull. Another distinction made is between the case of one observed solution [Ahuja and Orlin, 2001, Chan et al., 2019], and multiple observed solutions from either one [Burton and Toint, 1992b, Babier et al., 2019] or multiple [Aswani et al., 2018] feasible regions. Motivated by practical applications, we focus on proposing exact and heuristic algorithms for imputing the cost vector of a linear integer optimization problem given multiple imperfect observations originating from one known feasible region. We use "imperfect" to represent the integer error in the observations since the term "noisy" usually refers to the non-integer error.

The motivation for studying the setting with multiple observations for the same optimization problem is threefold. First, learning model parameters based on a single observation may lead to biased estimates of the costs if the observation is a measurement subject to noise. This situation is particularly relevant for physical/geoscientific applications where

"multiple noisy observations of different origin need to be combined to improve the reconstruction of a common underlying quantity" [Gerhards et al., 2017], and where "sometimes, due to strong noise (with unknown statistics), the measurement is not trivial" and "objectivity can only be attained if data redundancy is great enough that differences in data interpretation among different observers do not significantly alter the models obtained" [Tarantola, 2005]. Second, if we use IO to impute a cost function that represents expert knowledge, it is important to recognize that multiple experts solving the same underlying optimization problem will propose different solutions, and the task of imputing the costs will implicitly lead to a consensus [Troutt, 1995]. For instance, multiple dietitians may propose slightly different diets to a patient, despite having the same set of nutritional constraints in mind. Similarly, we may want to reconcile varying behavior of a decision-maker, such as when imputing one utility function of a consumer whose (multiple) transactions we can observe, under the setting of a fixed budget constraint. Finally, the multi-point, single feasible region setting is important as a sub-problem of more complex or computationally intensive problems. For instance, Babier et al. [2019] studied the inverse linear optimization problem with multiple observations and showed how to use it to aggregate the results of an ensemble of machine learning models for obtaining a good treatment plan for intensity-modulated radiation therapy.

In this chapter, we formulate the problem of imputing a cost vector given multiple, potentially imperfect observations that lie within one known feasible region. As such, this study addresses the *integer* programming analog of the recent work by Babier et al. [2019]. It is also an extension of a recent manuscript by Moghaddass and Terekhov [2019] (chapter 2), which solves the problem of imputing $\mathbf{c}$ for a single imperfect observation for the setting of integer linear optimization. For representing inverse integer optimization with multiple points, our formulations aim to minimize either the sum or the maximum of the optimality

gaps over all the observations. Due to the non-linearity of the resulting formulations, we propose a cutting plane algorithm and a linear-programming-based heuristic for solving the problem. Computational experiments compare the solutions of the specified inverse problem based on one observation with solutions based on multiple observations, as well as the performance of the exact cutting plane algorithm with that of a heuristic method. These results show a) the value of using multiple points to estimate the cost vector, b) that the cutting plane algorithm can effectively solve small-sized problems, and c) that the heuristic approach is an effective method for larger problems due to its ability to obtain near-optimal inverse solutions in a short amount of time.

3.3 LITERATURE REVIEW

As mentioned above, the most closely related literature includes papers on inverse integer optimization with a single point and inverse continuous optimization with multiple points. Our work establishes a bridge between these two areas by focusing on the inverse integer case with multiple points.

3.3.1 Inverse Integer Optimization with a Single Point

To date, the focus of the inverse integer optimization work has been on the single-observation case. Wang [2009] presented a cutting plane algorithm for finding the minimal adjustment of a cost vector of a mixed-integer linear problem so that a single observation is a candidate optimal. He showed that the cutting plane algorithm was useful for small mixed-integer linear problem instances from MIPLIB. Bulut and Ralphs [2016] studied the same problem, also proposing a cutting plane algorithm but focusing on the analysis of computational complexity. Schaefer [2009] provided two algorithmic approaches for solving the same inverse integer problem. The first approach is to find the integral hull of

the problem and apply inverse linear programming to it. The second approach is to provide a polyhedral description of the inverse-feasible objectives for a pure integer program. To overcome the limitations of the cutting plane algorithm, such as high CPU time required when solving large instances and the algorithm's inability to provide intermediate feasible solutions, Duan [2009] introduced a parallel algorithm which can also provide feasible solutions before termination. Our previous work in chapter 2 proposed an IO model and a cutting plane algorithm for a single *imperfect* integer observation.

3.3.2 Inverse Continuous Optimization with Multiple Points

In inverse continuous optimization, there is a substantial interest in the multi-point case, with emphasis on IO with multiple observations from related but distinct optimization problems [Esfahani et al., 2018, Aswani et al., 2018, Keshavarz et al., 2011, Bertsimas et al., 2015].

In two of the earliest applications of IO, i.e., seismic tomography and transportation planning, the goal was to find a minimal perturbation of the arc costs to make an observed set of paths optimal [Burton and Toint, 1992a, 1994]. These problems are special cases of the IO problem in which the input is a set of observations that represent potential solutions to one optimization problem. More recently, Babier et al. [2019] addressed the more general version of that problem, i.e., the problem imputing a cost vector based on multiple observations that are feasible or infeasible solutions to one linear (continuous) optimization problem.

Troutt [1995] addressed a parametric IO problem with multiple decision-makers and multiple observations for each decision-maker by developing a scoring model using the

approach of maximum decisional efficiency. Keshavarz et al. [2011] proposed an optimization model based on KKT optimality conditions for imputing objective function coefficients of a parametric convex optimization problem. Bertsimas et al. [2015] estimated the utility functions of players in a game by combining ideas of IO with the theory of variational inequalities using the player's observed actions. Furthermore, Aswani et al. [2018] worked on finding unknown parameters of continuous optimization models to make multiple noisy observations approximately optimal. Esfahani et al. [2018] applied parametric IO to predict the parameters of the objective function using incomplete information of observers. Saez-Gallego and Morales [2017] focused on energy consumption, estimating the unknown parameters to predict the demand for electricity according to changing the price at a different time to minimize the out-of-sample prediction error.

Based on the literature, to the best of our knowledge, there is no prior work focusing on using inverse optimization with multiple feasible integer observations from the same feasible region, which is our purpose of this chapter. These observations can be imperfect.

3.4 METHODOLOGY

In this section, we first present the forward integer optimization model and then generalize the inverse integer optimization model from previous work (chapter2) to the case of multiple observations. The goal is to impute a unit cost vector that makes the given set of feasible observations $\hat{\mathcal{X}} = \{\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \hat{\mathbf{x}}_3, ..., \hat{\mathbf{x}}_N\}$ optimal or approximately optimal solutions of the forward problem. We let $\mathcal{N} = \{1, \ldots, N\}$ denote the index set of $\hat{\mathcal{X}}$.

3.4.1 Forward Integer Optimization Model

Let $\mathbf{A} \in \mathbb{R}^{I \times J}$, $\mathbf{b} \in \mathbb{R}^{I}$, and $\mathbf{c} \in \mathbb{R}^{J}$ where I and J denote the number of constraints and decision variables, $\mathbf{c}'$ denotes transpose of the vector $\mathbf{c}$. The forward integer optimization

(FIO) problem is:

$$\mathbf{FIO}(\mathbf{c}): \underset{\mathbf{x}}{\text{minimize}} \quad \mathbf{c}'\mathbf{x} \tag{5a}$$

$$\text{subject to} \quad \mathbf{A}\mathbf{x} \geq \mathbf{b} \tag{5b}$$

$$\mathbf{x} \in \mathbb{Z}^J. \tag{5c}$$

For convenience, we define $\mathcal{X} = \{\mathbf{x} \mid \mathbf{A}\mathbf{x} \geq \mathbf{b}, \mathbf{x} \in \mathbb{Z}^J\}$ and $conv(\mathcal{X})$ as the convex hull of $\mathcal{X}$, also referred to as the integral hull. We assume $conv(\mathcal{X}) \neq \emptyset$ and is bounded. We let $\mathcal{X}^{\text{OPT}}(\mathbf{c})$ be the set of optimal solutions to $\mathbf{FIO}(\mathbf{c})$, $\mathcal{X}^{\text{OPT}} := \cup_{\mathbf{c} \neq \mathbf{0}} \mathcal{X}^{\text{OPT}}(\mathbf{c})$ and $\mathbb{S} \subseteq \mathcal{X}^{\text{OPT}}$ be the index set of all (integer) extreme points of $conv(\mathcal{X})$.

3.4.2 Inverse Integer Optimization Models

The goal of our IO model is to find a $\mathbf{c}$ that makes the given set of observations $\hat{\mathcal{X}} \subseteq \mathcal{X}$ optimal or approximately optimal solutions of $\mathbf{FIO}(\mathbf{c})$. To do so, we provide two formulations, both based on the notion that "error" induced by the observations can be captured via the *optimality gap*, defined as the difference between objective values of a given solution and an optimal solution. We minimize either the sum or the maximum of the optimality gaps over all the observations. Note that we can refer to these two alternatives as minimizing the 1-norm and the infinity-norm, respectively, of an N-dimensional vector of the optimality gap induced by each point in $\hat{\mathcal{X}}$; however, since we assume all observations are feasible, all optimality gaps are non-negative, eliminating the need for the absolute value and reducing to simply a sum or a maximum taken over all the errors, respectively.

Minimizing the Summation of Optimality Gaps The goal of model (6) is to minimize the summation of optimality gaps. As $\mathbf{FIO}(\mathbf{c})$ is a minimization problem, the optimality

gap e_n^* is the difference between the objective function of a given solution, $\mathbf{c}'\hat{\mathbf{x}}_n$, and an optimal solution, $\mathbf{c}'\mathbf{x}^*$. This gap is the greatest non-negative gap among all extreme points of $\mathcal{X}$, i.e., $\mathbf{c}'\hat{\mathbf{x}}_n - \mathbf{c}'\mathbf{x}^* \geq \mathbf{c}'\hat{\mathbf{x}}_n - \mathbf{c}'\mathbf{x}^s$, $\forall \mathbf{x}^s$ in the set of extreme points, $\mathbb{S}$. $\mathcal{K}$ denotes the index set of $\mathbb{S}$. By replacing $\max_{n\in\mathcal{N},s\in\mathcal{K}} e_n^s = z_n$ and denoting the vector of optimality gaps as $\mathbf{e}_n = \{e_n^1, \ldots, e_n^{|\mathcal{K}|}\}$, our integer inverse optimization (IIO) model can be written as follows:

$$\mathbf{IIO}_1(\hat{\mathcal{X}}) \ : \underset{\mathbf{c},\mathbf{e}_n,\mathbf{y}}{\text{Minimize}} \quad \sum_{n=1}^{N} z_n \tag{6a}$$

$$\text{subject to :} \quad \mathbf{A}'\mathbf{y} = \mathbf{c}, \mathbf{y} \geq \mathbf{0} \tag{6b}$$

$$\|\mathbf{c}\|_1 = 1 \tag{6c}$$

$$\mathbf{c}'\hat{\mathbf{x}}_n \leq \mathbf{c}'\mathbf{x}^s + e_n^s \quad \forall \mathbf{x}^s \in \mathbb{S}, n \in \mathcal{N} \tag{6d}$$

$$e_n^s \leq z_n \quad \forall s \in \mathcal{K}, n \in \mathcal{N} \tag{6e}$$

$$\mathbf{e}_n \geq \mathbf{0} \quad \forall n \in \mathcal{N} \tag{6f}$$

Constraint (6b) describes the dual feasibility of the LP relaxation of $\mathbf{FIO}(\mathbf{c})$. (6c) is a normalization constraint that prevents the trivial solution $\mathbf{c} = \mathbf{0}$. Constraint (6d) indicates that for each given observation $\hat{\mathbf{x}}_n$, its objective value of $\mathbf{FIO}(\mathbf{c})$ is lower than the objective values of all other extreme points. Since there are multiple observations, it is sometimes impossible to make all these solutions the lower bound or optimal solution of $\mathbf{FIO}(\mathbf{c})$ simultaneously. Hence, a non-zero optimality gap can exist for each given solution, and the goal is to minimize the function of these gaps.

Minimizing the Maximum Optimality Gap Model (7) minimizes the maximum of the optimality gaps.

$$\mathbf{IIO}_2(\hat{\mathcal{X}}): \underset{\mathbf{c},\mathbf{e}_n,\mathbf{y},Q}{\text{Minimize}} \quad Q \tag{7a}$$

$$\text{subject to}: \quad \text{(6b)}, ..., \text{(6f)} \tag{7b}$$

$$z_n \leq Q \quad \forall n \in \mathcal{N} \tag{7c}$$

Both models (6) and (7) are nonlinear and require an algorithm for implementing. While we leave complexity analysis of the problem for future work, we conjecture that the problem inverse integer optimization problem with noisy data is NP-complete, following Theorem 8 of Bulut and Ralphs [2016], which shows the NP-completeness of the inverse mixed-integer LP lower-bounding problem for the candidate optimal case with a prior $\mathbf{c}$. In the next section, we propose a cutting plane algorithm for finding solutions to this model. In Section 3.6, we present experimental results with both objective functions.

3.5 SOLUTION APPROACHES

We propose two solution methods: an exact cutting plane algorithm and a heuristic. Both are the extensions of our proposed methods in our previous work (chapter 2).

3.5.1 Cutting Plane Algorithm

Algorithm 2 shows our proposed cutting plane algorithm for multiple integer observations, where s represents the iteration number, and N represents the number of observations. In each iteration, the algorithm adds N constraints to the inverse model, each corresponding to one of the N given observations. The algorithm terminates whenever all points become optimal, or there is no further improvement for the objective function. This algorithm is applicable for both **IIO** models (6 and 7). In this algorithm, the variable *BestSum* refers to the best summation of errors which is the smallest summation so far, and *BestMax* denotes the best maximum error. Since, in each iteration, the algorithm adds N

valid inequalities to the inverse problem, the computational complexity of the inverse problem solved at each iteration increases. In this algorithm, t denotes the number of current iteration, and in each iteration, the algorithm adds one new extreme point to the set $\mathbb{S}$.

Algorithm 2 Cutting Plane for Inverse Integer Optimization with Multiple Observations

---

$t \leftarrow 0$, $\mathbb{S}' \leftarrow \emptyset$ ▷ Initialization
$\mathbf{c}^0 \leftarrow \mathbf{IIO}(\hat{\mathcal{X}}, \mathbb{S}')$
$\mathbf{x}^0 \leftarrow \mathbf{FIO}(\mathbf{c}^0)$
$\epsilon_n^0 = \mathbf{c}^{0\prime}\hat{\mathbf{x}}_n - \mathbf{c}^{0\prime}\mathbf{x}^0 \quad \forall n \in \mathcal{N}$
For model 6: BestSum $\leftarrow \sum_{n=1}^{N} \epsilon_n^0$
For model 7: BestMax $\leftarrow \max_{n\in\mathcal{N}} \epsilon_n^0$
$\mathbf{c}^{\text{best}} \leftarrow \mathbf{c}^0$
while $\mathbf{c}'^t\hat{\mathbf{x}}_n > \mathbf{c}'^t\mathbf{x}^t \quad \forall n \in \mathcal{N}$ **do** ▷ Optimality condition
    $\mathbb{S}' \leftarrow \mathbb{S}' \cup \mathbf{x}^t$
    $t \leftarrow t + 1$
    $\mathbf{c}^t \leftarrow \mathbf{IIO}(\hat{\mathcal{X}}, \mathbb{S}')$
    $\mathbf{x}^t \leftarrow \mathbf{FIO}(\mathbf{c}^t)$
    if (For model 6):$\sum_{n=1}^{N} \epsilon_n^t$ = BestSum, (For model 7): $\max_{n\in\mathcal{N}} \epsilon_n^t$ = BestMax, **then** :
        if $\mathbf{c}^t = \mathbf{c}^{\text{best}}$ **then**
            Stop
        else
            $\mathbf{c}^{\text{best}} \leftarrow \mathbf{c}^t$
        end if
    end if
    if (For model 6): $\sum_{n=1}^{N} \epsilon_n^t$ < BestSum , (For model 7): $\max_{n\in\mathcal{N}} \epsilon_n^t$ < BestMax, **then**
        BestSum $\leftarrow \sum_{n=1}^{N} \epsilon_n^t$ or BestMax $\leftarrow \max_{n\in\mathcal{N}} \epsilon_n^t$
        $\mathbf{c}^{\text{Best}} \leftarrow \mathbf{c}^t$,
    end if
end while
Output: $\mathbf{c}^* \leftarrow \mathbf{c}^{\text{best}}$, $\sum_{n=1}^{N} \epsilon_n^* \leftarrow$ BestSum Or $\max_{n\in\mathcal{N}} \epsilon_n^* \leftarrow$BestMax

---

3.5.2 A Heuristic Approach Based on LP-Relaxation

As an alternative to the cutting plane algorithm, we propose a heuristic method: solving the multi-point inverse optimization formulation of the linear programming relaxation of $\mathbf{FIO}(\mathbf{c})$. Models (8) and (9) state the two IO models for the LP-relaxation of $\mathbf{FIO}(\mathbf{c})$, the first of which was proposed by Babier et al. [2019] and the second differing only in the objective function choice. These models minimize the sum and max, respectively, of the absolute duality errors, ϵ_n^a, $\forall n \in \mathcal{N}$. That is, unlike in the models above, here ϵ_n^a represents the difference between the objective values of the given solution $\hat{\mathbf{x}}_n$ and an optimal *linear programming* solution. We denote a vector of $\boldsymbol{\epsilon}^a$ as follows $\boldsymbol{\epsilon}^a = \{e_1^a, \ldots, e_N^a\}$. Constraints (8b) and (8c) enforce dual feasibility and strong duality, respectively. Constraint (8d) is normalization constraint which also prevents the trivial solution $\mathbf{c} = \mathbf{0}$.

Although the inverse LP models come from the paper of Babier et al. [2019], to the best of our knowledge, this is the first time that this heuristic model and the comparison is applied to multi-point inverse integer optimization.

$$\mathbf{ILP}_1(\hat{\mathcal{X}}): \quad \underset{\mathbf{c},\boldsymbol{\epsilon}^a,\mathbf{y}}{\text{Minimize}} \quad \sum_{n=1}^{N} \epsilon_n^a \tag{8a}$$

$$\text{subject to}: \quad \mathbf{A}'\mathbf{y} = \mathbf{c}, \mathbf{y} \geq \mathbf{0} \tag{8b}$$

$$\mathbf{c}'\hat{\mathbf{x}}_n = \mathbf{b}'\mathbf{y} + \epsilon_n^a \quad \forall n \in \mathcal{N} \tag{8c}$$

$$\|\mathbf{c}\|_1 = 1 \tag{8d}$$

$$\mathbf{ILP}_2(\hat{\mathcal{X}}): \quad \underset{\mathbf{c},\boldsymbol{\epsilon}^a,\mathbf{y},E}{\text{Minimize}} \quad E \tag{9a}$$

$$\text{subject to}: \quad \text{(8b), (8c), (8d)} \tag{9b}$$

$$\epsilon_n^a \leq E \qquad \forall n \in \mathcal{N} \tag{9c}$$

The above models are linear and can be solved directly using a commercial solver such as CPLEX. The process of using these models to solve the multi-point inverse integer problem is described as follows.

1. Solve $\mathbf{ILP}_1(\hat{\mathcal{X}})$ (model (8)) or $\mathbf{ILP}_2(\hat{\mathcal{X}})$ (model (9)) and find a cost vector $\mathbf{c}^{LP}$.
2. Solve $\mathbf{FIO}(\mathbf{c}^{LP})$ (model (5)) to find an optimal integer solution $\mathbf{x}^*$.
3. Based on the available information ($\mathbf{c}^{LP}$, $\mathbf{x}^*$ and multiple observations $\hat{\mathbf{x}}_n$) calculate the difference between the objective function of the optimal solution and the given solutions (thus finding the optimality gaps of the integer solutions).
4. Calculate the average or the maximum of the n optimality gaps obtained in step 3.

In the next section, we present experimental results with both the cutting plane approach and the LP-based heuristic.

3.6 EXPERIMENTAL RESULTS

In this section, we compare the results of IO with multiple integer observations obtained from cutting plane (further referred to as Multi-Point Cu-Pl) and LP heuristic (Multi-Point Heur). Furthermore, we compare the results of IO based on multiple integer points with that of IO based on a single integer point (Single-Point Cu-Pl). We make comparisons based on the optimality gaps, computational time, and the number of iterations.

We implement the experiments for seven different problem sizes. For each size, we generate 20 instances; we consider 10 different interior points for each instance. In this study, we assume a unit weight for all optimality gaps. As future work, we can weigh the errors coming from different observations differently.

3.6.1 Comparing Optimality gaps Between Methods

In this part, we focus on comparing the *optimality gaps* obtained from different methods, i.e., Multi-Point Cu-Pl, Single-Point Cu-Pl, and Multi-Point Heur. We consider both minimizing the summation and minimizing the maximum of optimality gaps as the objective of the inverse models.

3.6.1.1 Minimization of the Sum of Optimality Gaps

In this part, we consider the IO model (6) with multiple integer observations when the objective is minimizing the summation of optimality gaps and use a cutting plane algorithm to solve it. Also, we take the IO model (8) and use the heuristic approach. Table 3.1 represents the corresponding results – all values shown are averages over 20 different instances for each of the seven problem sizes. Furthermore, we compare to using the cutting plane algorithm for solving IO with a single integer observation (from Chapter 2).

For Multi-Point Cu-Pl, given 10 observations, we find a unit cost vector and calculate the optimality gaps for 10 given points based on the obtained $\mathbf{c}$. Furthermore, using a Single-Point Cu-Pl method for inverse integer optimization [Moghaddass and Terekhov, 2019] (Chapter 2), we obtain 10 different cost vectors from each given point. We will find the optimality gaps using each of these obtained cost vectors (10 optimality gaps for each obtained cost vector). Then, we choose the smallest average of optimality gaps. Then, we compare this smallest average with the average of optimality gaps obtained from the Multi-Point Cu-Pl method.

According to the results of Table 3.1, the averages of optimality gaps for using Multi-Point Cu-Pl are always less than the smallest average of optimality gaps using a Single-Point Cu-Pl. In this table, all numbers are rounded to two decimals places.

Table 3.1 also demonstrates a comparison between the average of optimality gaps obtained from Multi-Point Cu-Pl and Multi-Point Heur methods. The results show that the solutions obtained from using the cutting plane algorithm and the LP heuristic are very close to each other.

The remaining columns compare the difference and relative errors between the optimality gaps obtained from columns number 1 & 2 and columns number 1 & 5.

Figure 3.1 summarizes the results of Table 3.1. In this figure, we see that the average of optimality gaps obtained from Multi-Point Cu-Pl is always a lower bound for the other methods; however, the Multi-Point Heur performs well and can be a good replacement for the exact Multi-Point Cu-Pl.

Size	**Comparing Average of Optimality Gaps**						
I x J	**1-** Avg. Opt-Gaps **Multi-Point Cu-Pl**	**2-** Smallest Avg. Opt-Gaps **Single-Point Cu-Pl**	**3-** Difference of Opt-Gaps	**4-** Relative Error %	**5-** Avg. Opt-Gaps **Multi-Point Heur**	**6-** Difference of Opt-Gaps	**7-** Relative Error %
4 x 2	5.96	7.56	1.60	23.53	5.98	0.02	0.35
5 x 2	9.68	13.02	3.34	39.04	9.72	0.04	0.32
20 x 5	1.42	1.87	0.45	30.20	1.42	0.00	0.00
40 x 5	2.04	2.55	0.51	34.89	2.04	0.00	0.00
50 x 5	3.40	4.00	0.60	20.40	3.40	0.00	0.00
35 x 10	0.42	0.48	0.05	12.23	0.43	0.00	0.00
90 x 10	0.47	0.53	0.06	11.87	0.47	0.00	0.05

Table 3.1: Comparing Average of Optimality Gaps for Summation of Gaps

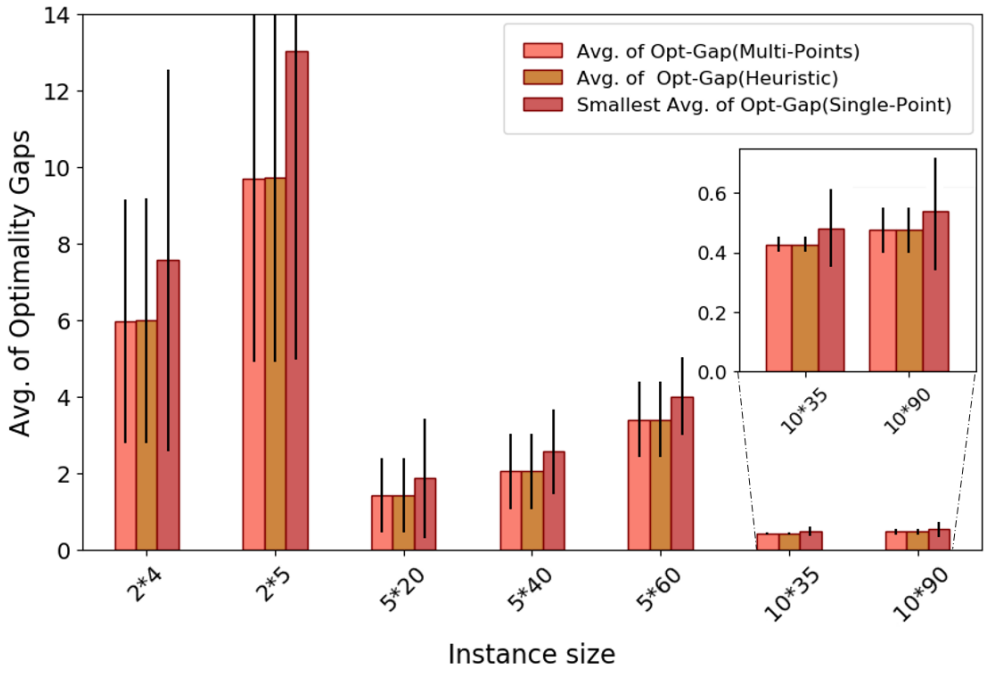


Figure 3.1: Average of Opt-Gaps When the Objective is Sum of Opt-Gaps

3.6.1.2 Minimization of the Maximum Optimality Gap

Here, since the objective is minimizing the maximum of optimality gaps, we apply the inverse models (7) for solving Multi-Point Cu-Pl and model (9) for Multi-Point Heur. The comparisons are among the obtained maximum of optimality gaps.

For the first type of comparison, we run the Single-Point Cu-Pl algorithm for 10 different observations separately, obtaining a cost vector for each one. We calculate the optimality gaps of all observations with respect to each obtained cost vector. Then, for each cost vector, we choose the maximum of the optimality gaps (over all observations). We consider the smallest among the maximum errors computed for each cost vector. On the other hand, we then use the Multi-Point Cu-Pl algorithm with the same 10 points. We obtain one cost vector and select the maximum optimality gap. Table 3.2 shows that this maximum optimality gap is always less than or equal to the smallest maximum obtained

from Single-Point Cu-Pl. We rounded the numbers of this table to 2 decimal points.

Table 3.2 also illustrates the results of the Multi-Point Heur method. We obtain a cost vector and optimality gaps based on that and find the maximum of them. Regarding having multiple points, the results show that maximum opt-gap using Multi-Point Cu-Pl is always less than or equal to max-opt-gap, which Multi-Point Heur obtains. Figure 3.2 summarizes the results for the average of optimality gaps for 20 different instances and 7 different sizes.

Size	**Comparing Average of Maximum Optimality Gap**						
I X J	**1-** Max Opt-Gap **Multi-Point Cu-Pl**	**2-** Smallest Max Opt-Gap **Single-Point Cu-Pl**	**3-** Difference of Max Opt-Gaps	**4-** Relative Error % for	**5-** Maximum Opt-Gap **Multi-Point Heur**	**6-** Difference of Max Opt-Gaps	**7-** Relative Error % for
4 X 2	18.07	21.06	2.99	32.40	18.15	0.08	0.69
5 X 2	29.39	31.77	2.38	10.16	29.42	0.04	0.11
20 X 5	3.86	7.02	3.15	115.98	3.87	0.01	0.26
40 X 5	6.12	10.27	4.15	103.86	6.13	0.01	0.33
60 X 5	9.41	14.51	5.09	57.53	9.42	0.00	0.05
35 X 10	0.49	1.02	0.52	104.47	0.50	0.01	1.28
90 X 10	0.64	1.24	0.60	77.20	0.64	0.00	0.07

Table 3.2: Comparing Maximum Opt-Gaps Between Different Methods

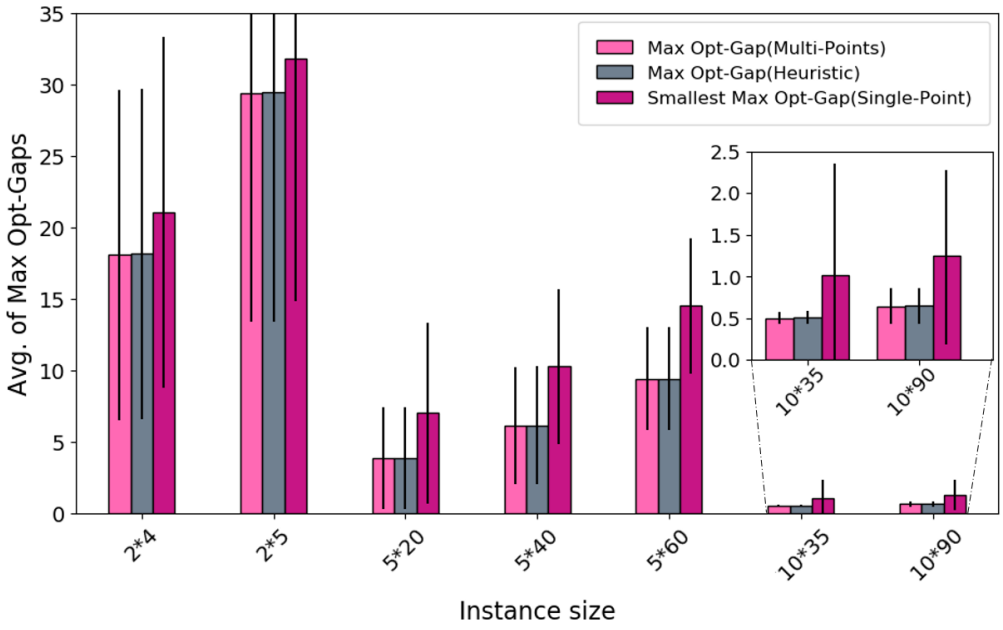


Figure 3.2: Maximum Opt-Gaps When the Objective is Min-Max

The results of Tables 3.1 and 3.2 are demonstrating that although cutting plane algorithm can always find an exact solution, an LP relaxation heuristic method works well and the solutions are close to the cutting plane.

3.6.2 Comparing Average of CPU Time and Average Number of Iterations Between Methods

In this part, we compare the averages of CPU time and the number of iterations between different approaches. Based on the results shown in Figure 3.3, employing multiple observations helps to find the solution sooner. If we use a single observation each time, we should solve the problem 10 times, and it will waste some time. The results compare CPU time for a Multi-Point Cu-Pl method with an average of CPU time using the Single-Point Cu-Pl. It demonstrates that CPU time for Multi-Point Cu-Pl methods(min-max and summation) is mostly less than an average of CPU time for a Single-Point Cu-Pl. Another conclusion obtained from the chart is that the algorithm works fast most of the time in the

small size of instances. We sometimes see surprising results, e.g., for an instance size with 10 by 30, we observe that there is a significant change where the CPU time for Multi-Point Cu-Pl is higher than the average CPU time for a Single-Point Cu-Pl. Also, regarding the heuristic lines, it is evident that Multi-Point Heur can find the solution sooner without any iterating.

Figure 3.4 shows the number of iterations for finding a cost vector using Single-Point Cu-Pl and Multi-Point Cu-Pl methods(based on two different objectives). It also compares them with the average number of iterations for finding 10 cost vectors for Single-Point Cu-Pl and also the Multi-Point Heur. The results are a summary of the average results of 20 instances for each size. The line chart illustrates that Multi-Point Heur can find the solution with 0 iterations. Also, the number of iterations for finding a cost vector using Multi-Point Cu-Pl is mostly less than the average number of iterations using a Single-Point Cu-Pl.

Using different experiments, we aim to show that the CPU time and the number of iteration depend entirely on the complexity of the forward problem. Hence, there may be other instances that do not work in the same pattern in terms of CPU time and the number of iterations. Therefore, these results are different from one case to another. We analyzed the complexity of the problem in chapter 2. As a summary, the complexity of the cutting plane algorithm depends on the complexity of the separation problem, which means that if the forward problem is hard to be solved based on the obtained cost vector, the consummation time for the algorithm increases. As future work, we can analyze the efficiency of the algorithm for large scale problems.

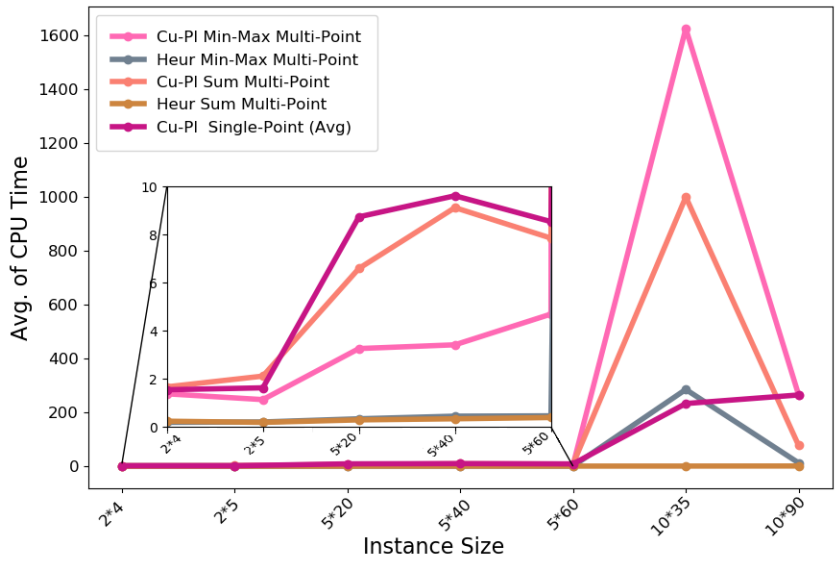


Figure 3.3: Comparing Average of CPU time Between Different Methods

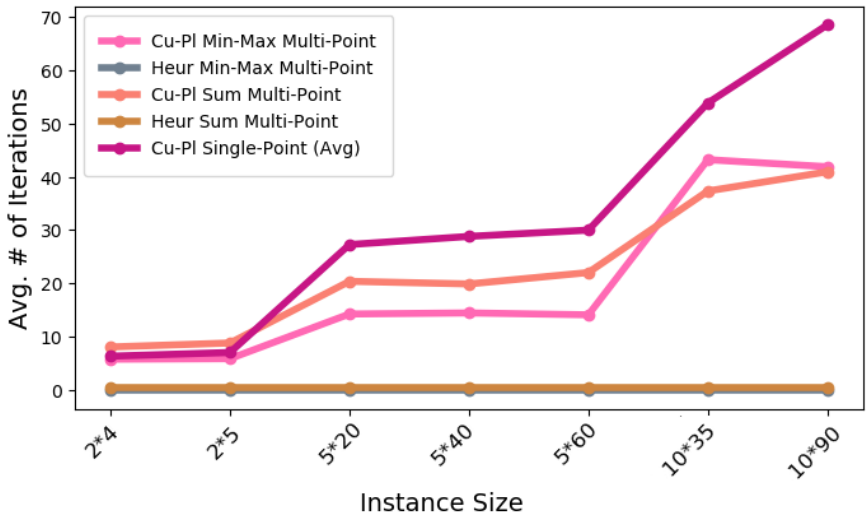


Figure 3.4: Comparing Average Number of Iterations Between Different Methods

3.7 DISCUSSION

In this section, we analyze the solutions of inverse integer optimization, focusing on the optimality gaps for multiple observations. By optimality gaps, we mean the difference

between the objective values of the given observations and optimal solution of the forward integer problem. We illustrate different scenarios that may happen depending on the location of the points (observations). We assume a linear optimization problem with two dimensions and will show that even if the points are on the boundary, the solution can be imperfect. We already explained some other cases in the previous chapter (chapter 2). In that work, we analyzed different scenarios that may happen for a single imperfect integer point. We explained why the solution of the heuristic is close to the solution from the cutting plane method. We also illustrated that the algorithm would give a smaller optimality gap rather than the heuristic method; however, the differences are not significant. In the current section, we focus on new scenarios for multiple points.

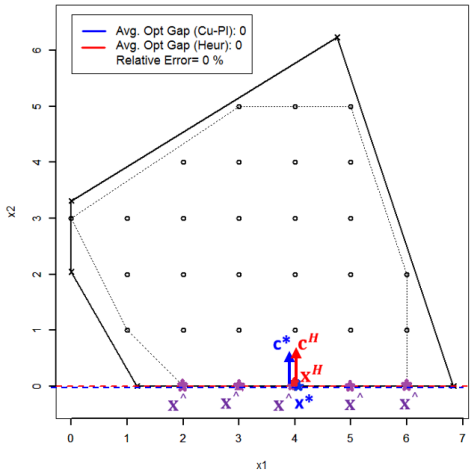
Before starting, we mention some relevant results which exist in the literature. In particular, Babier et al. [2019] stated that a given error within a larger feasible region indicates a better fit than the same error within a smaller feasible region. Also, the error in the larger feasible region is closer to the optimality in terms of goodness of fit. They proved that the closer multiple points are to each other, the fit is better.

3.7.1 Multiple Points on the Boundary of Integer Convex Hull

Figure 3.5 shows that if there are multiple given points which are feasible, on the same hyperplane, and on the boundary of the integral hull, there is a cost vector that can make all of these points optimal to the problem. The cutting plane algorithm and heuristic methods can both find an optimal solution. The optimality gaps for all of these points are equal to zero. The obtained cost vector is perpendicular to the hyper-plane where all points located.

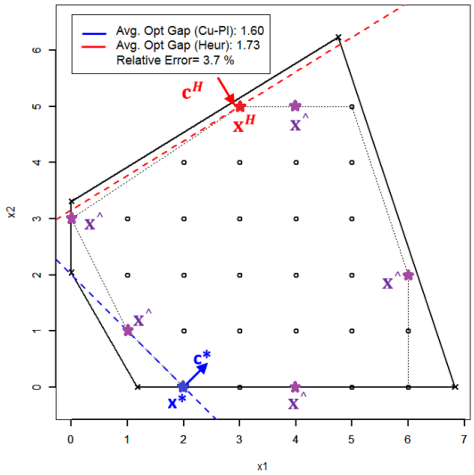
In contrast with Figure 3.5, where all points are on the same hyperplane, sometimes there are multiple given points, and each one of them lies on a different hyperplane. Although the given points are all on the boundary, it is impossible to find a cost vector that

makes all these points optimal at the same time. In other words, each of these observations is individually optimal, but considering them together implies the presence of error when finding one cost vector. Therefore, we consider optimality gaps, and the goal is to find the best cost vector that makes the points a candidate or close to the optimal. As Figure 3.6 shows, based on the obtained cost vector from both the heuristic and cutting plane algorithms, not all integer points are a candidate optimal. Therefore, this type of problem still falls into the class problem that we consider, i.e., inverse integer optimization with multiple imperfect observations. However, the figure well illustrates that even in this case, a cutting plane algorithm finds a better cost vector than the one obtained from the heuristic since the average of optimality gaps for cutting plane is smaller.



$\mathbf{c}^*$=(0,1), $\mathbf{c}^H$=(0,1)

Figure 3.5: Points on the Same Hyperplane



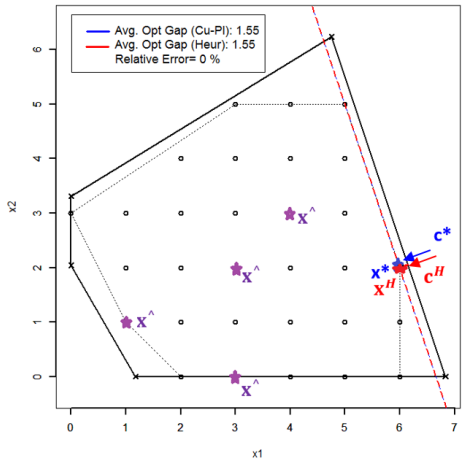
$\mathbf{c}^*$=(0.5, 0.5), $\mathbf{c}^H$=(0.38, -0.62)

Figure 3.6: Points on Different Hyperplanes

3.7.2 Multiple Points Inside and on the Boundary of Integer Convex Hull

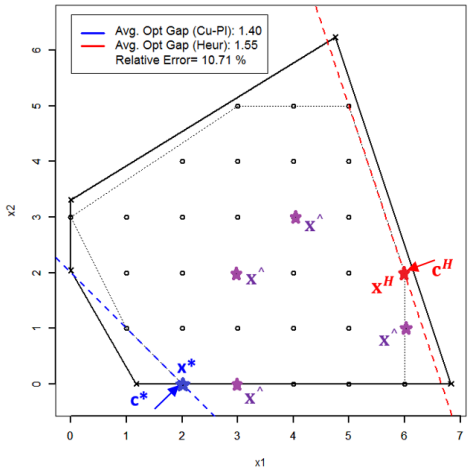
Figures 3.7 and 3.8 are illustrating that when the set of given points includes some points on the boundary and some inside the feasible region, it is not guaranteed that the obtained cost vector makes even the given points that are on the boundary optimal. While Figure 3.7 shows a case that both the heuristic and cutting plane algorithms find the same

result, Figure 3.8 demonstrates the case where the obtained cost vectors from the two methods are different from each other. However, the average of optimality gaps using a cutting plane algorithm is always smaller than or equal to the one using a heuristic method.



$\mathbf{c}^*$=(-0.75, -0.25), $\mathbf{c}^H$=(-0.75, -0.25)

Figure 3.7: Points Inside and on the Boundary (1)

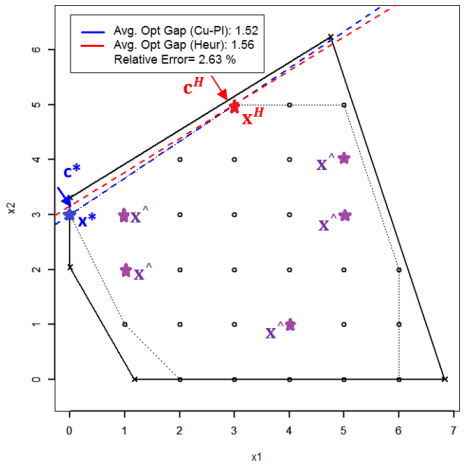


$\mathbf{c}^*$=(0.5, 0.5), $\mathbf{c}^H$=(-0.75, -0.25)

Figure 3.8: Points Inside and on the Boundary (2)

3.7.3 Multiple Interior Points Close to the Boundary of Integer Convex Hull

Figure 3.9 shows that the given points that are very close to the boundary. The average of optimality gaps for cutting plane is less than heuristic. However, the results of two approaches are very close to each other since the cost vectors are pointing in almost the same direction.



$\mathbf{c}^*$=(0.4, -0.6), $\mathbf{c}^H$=(0.38, -0.61)

Figure 3.9: Points Close to the Boundary

3.8 SUMMARY AND CONCLUSION

The purpose of an IO model is to estimate the unknown model parameters of a forward optimization problem using some data from the past. In a classical definition of IO, the observations were assumed to be candidate optimal solutions of the forward problem. However, recent work has shifted toward IO with multiple, imperfect observations where the goal is to estimate model parameters that minimize some notion of error for multiple observations. To the best of our knowledge, this is the first work to propose an IO model for *multiple imperfect integer* observations. We used "imperfect" instead of noisy for the existence of integer error. We proposed a cutting plane algorithm for solving the model. In this technique, the algorithm in each iteration adds multiple feasible generated cuts to the inverse problem. Although having multiple observations makes the problem more challenging, it has some advantages. In particular, doing so will be a guarantee for decision-makers to make the right decision for forecasting parameters for the future. In this study, we compared the results of cutting plane algorithm for multiple points and a

single point considering the average of optimality gaps, the number of iterations, and CPU time. Also, we proposed an LP heuristic with multiple points and showed the practicality of both approaches for multiple observations. As future work, we generalize the models and algorithms of this study for using a real application and comparing the results with other data-driven techniques.

Chapter 4

Data-Driven Inverse Optimization for Recommender Systems

4.1 ABSTRACT

An inverse optimization problem aims to identify the unknown optimization model parameters using past data and is a valuable method for prediction. Mainly, it can be useful when the goal is to predict user preferences or behaviors if the users are optimizing their choices. Given past users' preferences, the recommender system is a technique that can identify the relationships between user and item attributes to recommend the best items to the users. There are many studies on recommender systems in the literature, most employing machine learning methods. The contribution of this work is the application of inverse optimization as a new technique for recommender systems when the assumption is that users are optimizing their decisions; the existence of this assumption implies that less data is necessary for training, as compared to other methods for recommender system problems. Using a cutting plane algorithm, the results of a real case data-set for recommending the best restaurants to the customers demonstrate that inverse optimization can predict user's

choices with a precision of more than 70 % most of the time. This study also proposes a heuristic method as an efficient and reliable alternative for the cutting plane algorithm when the computational time is considerable.

4.2 INTRODUCTION

In a typical forward optimization problem, there are some model parameters. Given these parameters, the objective is to find an optimal solution to the problem and optimize the objective function. In practice, some parameters can be hard to obtain or estimate. However, if past observed solutions (i.e., prior decisions by optimizing agents) are available, it is possible to use them to identify the unknown model parameters. Given such data, inverse optimization (IO) is a technique that imputes the unknown model parameters to make the given observations optimal.

In Chapter 2 of this thesis, we proposed a model for solving an IO problem with an imperfect integer observation. Then, in Chapter 3, we investigated the situation where there are multiple integer observations instead of a single one. In this chapter, we apply the idea of IO with multiple (imperfect or optimal) integer observations as a technique for recommender systems (RS).

The RS technique is to recommend the best items to users where there are multiple items available, and also the attributes of users and items are accessible. Given multiple integer observations, which are prior users' ratings of the items, the goal is to find a user/item attribute weight matrix. The elements of this matrix are considered as coefficients of the objective function to make the given observations optimal or approximately optimal.

As already mentioned in Chapter 2, in a classical definition of IO, the researchers intended to impute the unknown model parameters that make the given observation precisely optimal. However, this approach does not work when there is an error in the given data.

This error arises from two sources: one is the sub-optimality of the observations. In linear optimization, it means that one or some of the observations are an interior point of the feasible region of the forward problem. The second source of the "error" is the existence of multiple points that cannot be simultaneously optimal, even if each point is an optimal solution under a different set model coefficients. In linear programming, this case occurs, for instance, when all given points are on the boundary but different hyperplanes, which is impossible to find a cost vector that makes all points simultaneously optimal. Hence, the modern definition of IO considers the gap in optimality as a way to measure the error in the data, and we adopt the same idea in this chapter. We refer to integer data that are not candidate optimal points as *imperfect* data.

We develop a customized IO model for recommender systems which recommend the best items to the users based on the past users' ratings and relationships between the attributes. In Chapter 3, we assumed that all observations are solutions to one optimization problem, and there is a single feasible region for them. In the current work, each user has a different constraint set; thus, we assume that each observation in the data set comes from a different feasible region.

4.2.1 What Is a Recommender System?

Recommender systems that have become increasingly popular in recent years are practical in a variety of areas, including recommending movies (e.g., Netflix), music (e.g., YouTube), and other products (e.g., Amazon). There are also RS for experts [Chen et al., 2015], collaborators [Chen et al., 2011], and financial services [Felfernig et al., 2007]. Recommender systems are an important part of the information and e-commerce ecosystem [Ekstrand et al., 2011]. Netflix and Amazon are among the two most visited websites with more than 17,000 movies and 410,000 items, respectively, available on their websites.

Decision making about what to watch or buy among the variety of items is challenging, even with the help of friends, reviews, or expert opinion. As a solution, computer-based systems provide the opportunity to expand the set of people from whom users can obtain recommendations [Ekstrand et al., 2011]. RS recognizes a way to predict *ratings* that a user gives to an item [Ricci et al., 2011] with the aid of past users' data and similarities between the users or items.

According to a definition by Vargas-Govea et al. [2011], RS is an online service that helps users to overcome the overload of data by retrieving useful items according to their ratings. Recommender systems predict the behavior of the users using past data. In an online system for choosing the best items for a user where the users sign up to the system, they should represent their attributes. Moreover, the item attributes and ratings of the past users are available in the system. Using this information, RS can recommend the best items to the user. RS must interact with the user, both to learn the user's ratings and to provide recommendations [Ekstrand et al., 2011]. Figure 4.1 illustrates the process, as well as the input and output of recommender systems. As shown in the figure, RS recommends items to any user (new or past user) based on the ratings of prior users and the attributes of users and items.

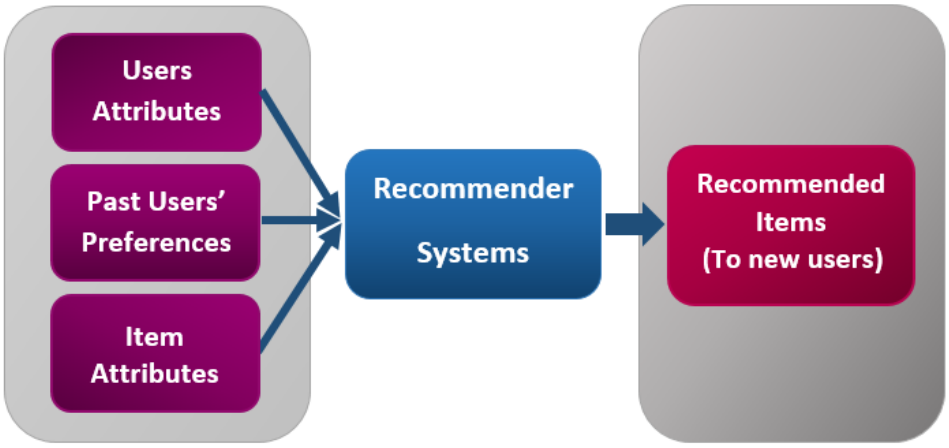


Figure 4.1: Recommender Systems Process

4.2.2 Literature Review

This literature review focuses on reviewing work in RS. The main approaches in the literature for RS include content-based filtering, collaborative filtering, and other techniques such as decision trees, association rules, and semantic approaches [Jiang et al., 2010].

Collaborative Filtering Collaborative filtering (CF) is a technique of RS that recommends the best items to the users based on the similar ratings that other users have expressed for those items. Amazon.com and Netflix have been using collaborative filtering for a decade to recommend products to their customers [Ekstrand et al., 2011]. There are two approaches for CF: user-based and model-based [Deshpande and Karypis, 2004]. A user-based approach refers to a method that leverages the similarities within a group of users (i.e., each person belongs to a larger group with similar behavior). Hence, the items purchased by other members of the group can be applied to form a basis to recommend items to the other members. On the other hand, a model-based approach analyzes historical data to identify the relationship between items. It assumes that buying a set of items often leads to purchasing other items. It recommends items, after finding the relationships

between them.

User-based CF has some limitations. One is its difficulty in measuring the similarities between users, and the other is the scalability issue. As the number of customers and products increases, the computation time of algorithms grows exponentially [Jiang et al., 2010]. Using a user-based collaborative filtering method needs dense data sets, and it needs at least a few people that have already evaluated a product. It does not reflect uncertainty in predictions and provides few if any, reasons for a recommendation [Ansari et al., 2000]. The model-based CF overcomes the scalability problem as it calculates item similarities from an offline database. It assumes that a user will be more likely to purchase items that are similar or related to the items that he/she has already purchased. Therefore, the model-based algorithms identify a user/item matrix to find the relationships between the items and use these relations for recommendations [Deshpande and Karypis, 2004].

Content-based Filtering The content-based filtering (CBF) method applies content analysis to target items. Target items are described by their attributes, such as color, shape, and material; recommender systems recommend based on the item attributes. The system builds the user's profile by analyzing his/her responses to questionnaires, his/her rating of products, and navigation history [Jiang et al., 2010]. The recommendation system proposes items that have high correlations with a user's profile. However, using this method, users can only receive recommendations similar to their earlier experiences. Some machine learning techniques will apply for CBF, such Bayesian classifiers, cluster analysis, decision trees, and artificial neural networks [Blanda, 2016].

Other Methods Based on CF and CBF, some studies developed new data mining techniques using decision trees, association rules, regression models, and Markov chains. The

goal was to recommend movies and books, support one-to-one online marketing, and attract customers for the tourism industry [Jiang et al., 2010]. In Vargas-Govea et al. [2011], the authors utilized machine learning techniques for recommending restaurants. Reviewing this paper resulted in the inspiration for our work. Their study states that common recommendation approaches take into account only user-item-rating data, ignoring contextual information. They apply contextual features in their work to improve user satisfaction by getting more knowledge about users. Contextual features refer to the user attributes and information that are important to know, such as location and time. To reduce the dimension of the problem were identified only relevant contextual features using a feature selection method. These feature selection techniques contributed to improving the efficiency of the contextual RS. After identifying the best features, the next step was analyzing their effects on different aspects of the system's performance. This paper showed that applying the reduced subset of attributes did not reduce the system performance, and using the feature selection technique made the contextual recommender system more efficient. Baltrunas and Amatriain [2009] proposed a contextual pre-filtering technique for recommendation called micro-profiling. In this paper, the authors had a long-term approach that presents time-aware recommender systems for predicting user's music tastes. Instead of using a single profile of users, they split it into multi-micro profiles for different periods for increasing the accuracy of the prediction. Each of these time slices represents the similar repetitive behavior of the user. They believed that each user has different behavior at different times, like morning, evening, and weekend. They evaluated methods for different time splits and demonstrated that using micro-profile data can increase the accuracy of the algorithm.

Inverse Optimization Approach, in Contrast with Existing Methods There are several differences between IO and the other data-driven techniques as methods for recommender systems:

- In IO, the assumption is that users are optimizing their decisions. Hence, imputing model parameters leverages the optimality property of the given solutions.
- In contrast to machine learning techniques which require a large amount of data to be able to reliably solve the problem, using IO, even with a small amount of data, the predictions are reliable (under the assumption that the users are optimizers).
- There is no requirement to have some ratings for the new users to recommend them the best items. In IO, we assume that there is no information about new users' ratings for the items. However, most of the previous techniques could not recommend if there were no item ratings available for the new user.

4.2.3 Why Inverse Optimization for Recommender Systems?

IO is a process of finding unknown model parameters of a forward problem using past observations. In the literature of IO, there are several papers when the observation is integer and a candidate optimal, namely by Wang [2009], Bulut and Ralphs [2016], Schaefer [2009]. Using a single past observation, IO found the unknown coefficients of the objective function to minimize the deviation between a prior cost vector and a cost vector that would make the given observation optimal.

In this chapter, we consider a forward IO problem that is a special case of an equality-constrained knapsack problem. A typical $\{0, 1\}$ knapsack problem is a linear integer programming problem that has just one inequality constraint for satisfying the capacity. There is also another form of this problem with an equality constraint called an equality knapsack problem (EKP) with $\{0, 1\}$ variables [Ram and Sarin, 1988]. There are some papers in the literature by Burkard and Pferschy [1995], Huang [2005], Roland et al. [2013] that considered the knapsack and the inverse of the knapsack problem. An article by Roland et al. [2013] discussed a $\{0, 1\}$ knapsack problem with a focus on identifying the minimal

adjustment of the profit vector in order to make the given feasible set of items become an optimal solution using a bi-level linear programming formulation. Also, a paper by Burkard and Pferschy [1995] addressed the case when the profit coefficients of the knapsack problem depend on a parametric function, and the goal was to find the smallest parameter (t^*) to make the optimal solution equal to specified solution value. Another paper by Huang [2005] demonstrated that a pseudo-polynomial algorithm could solve the inverse of the knapsack problem. The difference between inverse IO for this chapter and the inverse of the knapsack problem in the literature is that in this chapter, we assume that there are *multiple* observations coming from *different* feasible regions; furthermore, we allow the observations to be imperfect. However, in the literature of the inverse knapsack problem, the papers considered a *single candidate optimal* observation. In summary, the current algorithms available in the inverse knapsack problem literature would not be directly applicable to the recommender system problem of interest in this chapter.

Based on the literature, some studies focus on multiple noisy continuous observations such as the papers by Esfahani et al. [2018], Aswani et al. [2018], Keshavarz et al. [2011], Bertsimas et al. [2015]. However, to the best of our knowledge, there is no work focused on IO with multiple *imperfect integer* observations. The term "imperfect" here refers to the potential presence of *integer* error in the observations. We proposed a model with multiple imperfect integer observations in the previous chapter, Chapter 3.

In this chapter, we focus on RS and customize the IO problem for RS. There are many applications suggested for IO, including predicting the movements of earthquakes [Tarantola, 1987], healthcare therapy [Chan et al., 2014, Boutilier et al., 2015], transportation [Bertsimas et al., 2015], economics [Keshavarz et al., 2011, Bertsimas et al., 2015, Saez-Gallego and Morales, 2017], energy [Turner and Chan, 2013], scheduling [Li et al., 2013] and production planning [Troutt et al., 2008, 2006]. After reviewing different applications,

to the best of our knowledge, there is no work in IO with a focus on RS. Figure 4.2 shows our proposed schema for how IO would be applicable for recommender systems.

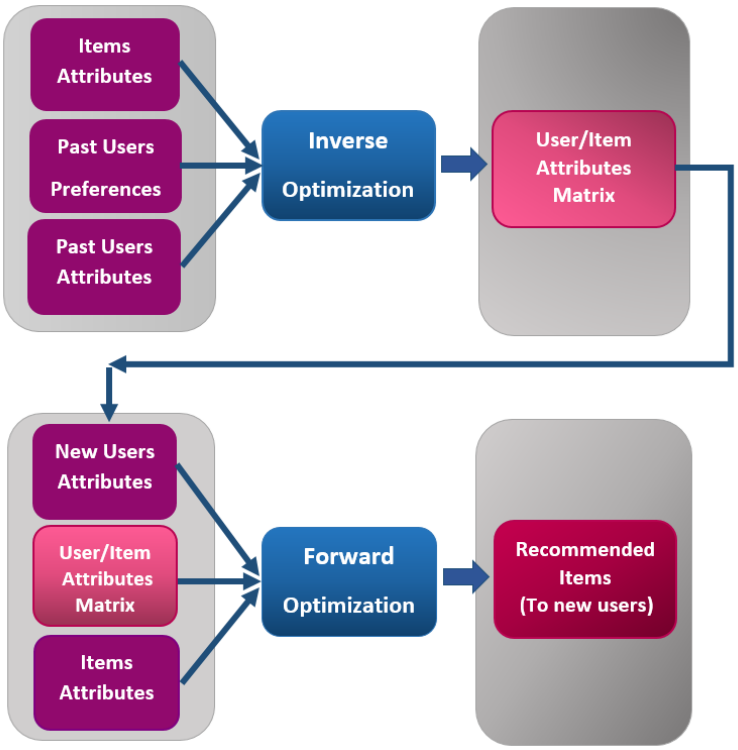


Figure 4.2: Inverse Optimization in Recommender Systems

In RS, the system requires to learn the user/item attribute weight matrix. Each element of this matrix demonstrates the importance of a user-attribute to an item-attribute. For instance, for the problem of choosing the best restaurants for users, if a user is senior and the item attribute is the availability of a parking lot in a restaurant, the element of the matrix defines the importance of having parking for a senior user. For example, if the weight element is 0.7, it means having parking for a senior is essential, and if it is 0.1, it means it is less important. Therefore, each element considers the importance of one user attribute to one item attribute. Since this matrix (or any prior knowledge about it) is unknown, we use IO to identify this matrix utilizing the ratings of past users, and user and item attributes.

After receiving this matrix from an inverse problem, it is used in the forward problem to recommend the best items to any user needing a recommendation.

We recommend IO as a technique for RS to predict the ratings of users. Using IO, not only can we find out the relationships between the attributes of users and items that affect the user decisions, but also we can recommend the best items to any user. We suggest the IO method when the assumption is that users are optimizing their choices. Moreover, in IO, we can find the solutions even with one or two points; however, machine learning techniques need a massive volume of information about past user ratings for the recommendation.

4.2.4 Contributions

The problem studied in this chapter is a particular case of IO with multiple integer observations where the observations are binary. We assume that the attributes for the users (past and new) are available. The contributions of this work are as follows:

- We focus on IO with the application in RS under the assumption that the users are optimizing their choices. To the best of our knowledge, there is no prior work on applying IO to RS problems.

- We develop a customized IO model for RS. IO assumes that users are optimizing their decision. Hence, we can leverage this assumption by using an optimization model; as a consequence, the amount of data required for imputing objective function parameters is substantially smaller as compared to typical (usually machine-learning-based) methods for recommender system problems.

- We propose a cutting plane algorithm and a heuristic method for solving IO in RS using multiple observations.

- We evaluate our proposed methods on real data sets for a restaurant recommender system. We evaluate both the quality of the predictions and computational times.

The organization of this paper is as follows. In Section 4.3, we propose the methodology and algorithm for IO in RS. Section 4.4 focuses on a case study that is applying IO in RS for choosing the best restaurants, and the real data obtained from different users are analyzed. Section 4.5 analyzes the results based on different sets of experiments. Section 4.6 discusses some cases in detail. Section 4.7 presents some future work ideas, and the chapter is summarized in Section 4.8.

4.3 METHODOLOGY

In this section, we propose a multi-point inverse integer optimization approach for RS. In the problem of RS, there are N users and M items. Users are people who have some ratings for items obtained in the past or for whom a recommendation needs to be made. Items are any kinds of products or services in an online system. We consider a separate forward optimization problem for each user. The forward optimization problem obtains the optimal ratings of the items for the user by maximizing the item values. There is a matrix of attribute weights in the objective function. It presents the relationships between the attributes of the users and items. However, obtaining the values of this matrix is not trivial. Since this matrix is unknown, we employ an IO problem to find it. IO uses prior data, which are past users' ratings, to impute the unknown matrix. The system obtained the information when the users signed up on the related online system. Getting this matrix will lead to predicting the item ratings for each user. After describing mathematical modeling for the forward and inverse optimization problem, we develop a cutting plane algorithm for solving the IO models. Then, we propose a heuristic approach to reduce computational time.

4.3.1 Forward Integer Optimization Problem for a Recommender System

In this part, we explain the forward integer optimization for RS (**FIOR**). **FIOR** is unique for each user and obtains the user's ratings. The feasible regions of **FIOR** problems are different from one user to another. Below, we mathematically define the forward optimization model for a single user.

Parameters

N: Total number of users.

M: Total number of items.

P: Total number of user attributes.

J: Total number of item attributes.

R_n: Number of items that should be recommended to user n.

$\mathbf{A}$: The user/item attribute weight matrix.

a_{pj}: The importance of user attribute p to item attribute j.

u_{np}: User attribute $\in \{0, 1\}$, i.e., 1 if user n has the attribute p and 0 otherwise.

h_{jm}: Item attribute $\in \{0, 1\}$, i.e., 1 if item m has attribute j and 0 otherwise.

v_{nm}: $\in \{0, 1\}$, i.e., 1 if a past user n has previously visited item m and 0 otherwise.

y_n: Dual variable related to constraint n in linear programming relaxation of the forward problem.

Sets

$\mathcal{N}$: Set of all users $\{n = 1, \ldots, N\}$

$\mathcal{M}$: Set of all items $\{m = 1, \ldots, M\}$

$\mathbb{S}_n$: Set of all extreme points in the feasible region of the forward problem of a particular user n.

$\mathcal{K}$: The index set of $\mathbb{S}$.

$\hat{\mathcal{X}}$: Set of all past observations $\{\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \hat{\mathbf{x}}_3, \ldots, \hat{\mathbf{x}}_N\}$ (one for each user n).

Decision Variables

x_{nm}: 1 if item m is assigned to the user n and 0 otherwise.

We can now formulate the forward optimization model for user n as follows:

$$\mathbf{FIOR_n}(\mathbf{R_n}) : \underset{x_{nm}}{\text{Maximize}} \quad \sum_{m=1}^{M}\{\sum_{p=1}^{P}\sum_{j=1}^{J} u_{np}a_{pj}h_{jm}\}v_{nm}x_{nm} \tag{10a}$$

$$\text{s.t. :} \quad \sum_{m=1}^{M} x_{nm} = R_n \tag{10b}$$

$$x_{nm} \in \{0,1\} \quad \forall m \in \mathcal{M}. \tag{10c}$$

Model (10) aims to choose the combination of items that maximizes the total preference value of the chosen items. Variable x_{nm} is a binary variable that represents whether the model assigns item m to the user n (1) or not (0). The constraint (10b) ensures that summation of the selected items for user n should be equal to R_n. By replacing $u_{np}a_{pj}h_{jm} = f_{nm}$, the objective function of the above model can be rewritten as $\underset{x_{nm}}{\text{Maximize}} \sum_{m=1}^{M} f_{nm}v_{nm}x_{nm}$. As we already mentioned in 4.2.3, the problem (10) is a special case of the equality $\{0,1\}$-Knapsack problem in which the weight of each item is 1. The value of each item, $f_{nm}v_{nm}$, is > 0 if the user n would like item m at some level and has previously obtained this item. Following the previous example, if the user is a senior ($u_{np} = 1$), the restaurant has parking ($h_{jm} = 1$), and the importance of parking to a senior (a_{pj}) is 0.7, then the value of this item ($u_{np}a_{pj}h_{jm}v_{nm}$) is 0.7 if the user has already visited this restaurant (obtained this item). We note that when we will be using this formulation for making recommendations to future users, the values of v_{nm} for all $m \in \mathcal{M}$ will be set to 1. Although in this thesis we use CPLEX to solve the above forward problem, an alternative approach is to simply sort the items according to their objective function coefficients and pick the top R_n since from a knapsack perspective, this case is equivalent to having items of equal weight [Skiena,

1997].

As future work, we can add some other constraints to the model (10). For instance, if the distance from the user's home and a restaurant is far, it would be automatically omitted from their choices. We can consider time or km for the distance criteria.

4.3.2 Multi-Point Inverse Integer Optimization Problem

Consider the situation that the objective function parameters a_{pj}, which represent the importance of user attribute p to item attribute j, are unknown for all p and j in $\mathbf{FIOR}$; in other words, we need to determine the elements of the matrix $\mathbf{A} = (a_{pj}) \in \mathbb{R}^{P \times J}$. We can learn the elements of this matrix with IO. Once $\mathbf{A}$ is estimated based on past user ratings, we can use the forward problem, instantiated with the estimated $\mathbf{A}$, to recommend items to any user.

Given a set of past observations for the $\mathbf{FIOR}$ problem for N users, $\{\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \hat{\mathbf{x}}_3, ..., \hat{\mathbf{x}}_N\}$ (each being a a feasible solution to the corresponding $\mathbf{FIOR}_n$ problem), the multi-point inverse integer optimization problem with the objective of minimizing summation of optimality gaps ($\mathbf{MIIO-SUM}$) is formulated as follows:

$$\mathbf{MIIO-SUM_1}(\hat{\mathcal{X}}): \underset{a_{pj}, e_n^s, c_n, y_n}{\text{Minimize}} \quad \sum_{n=1}^{N} \text{Max}_s\{e_n^s\} \tag{11a}$$

$$\text{s.t.} \quad y_n \geq f_{nm} v_{nm} \qquad \forall n \in \mathcal{N}, m \in \mathcal{M} \tag{11b}$$

$$\sum_{m=1}^{M} f_{nm} v_{nm} \hat{x}_{nm} + e_n^s \geq \sum_{m=1}^{M} f_{nm} v_{nm} x_{nm}^s \quad \forall \mathbf{x}_n^s \in \mathbb{S}_n, n \in \mathcal{N} \tag{11c}$$

$$f_{nm} = \sum_{p=1}^{P} \sum_{j=1}^{J} u_{np} a_{pj} h_{jm} \quad \forall n \in \mathcal{N}, m \in \mathcal{M} \tag{11d}$$

$$\sum_{p=1}^{P} \sum_{j=1}^{J} |a_{pj}| = 1 \tag{11e}$$

$$\|\mathbf{f}_n \mathbf{v}_n\|_1 = 1 \quad \forall n \in \mathcal{N} \tag{11f}$$

$$e_n^s \geq 0 \qquad \forall n \in \mathcal{N}, s \in \mathcal{K} \tag{11g}$$

In this model, in (11b) y_n is the dual variable corresponding to constraint (10b) if we constructed the dual of the LP relaxation. The motivation for this constraint was based on the intuition from the other **IIO** models in previous chapters, but the results would be most probably the same without it. Further investigation should be done to determine whether it can be eliminated. Constraint (11c) illustrates that for each user, the objective value of the **FIOR** model (10) for each given solution $\hat{\mathbf{x}}_n$ should be greater than or equal to the objective values of all other extreme points of the model (10). This constraint guarantees that $\hat{\mathcal{X}}$ is optimal or close to the optimal. Model $\mathbf{FIOR_n}$ is a maximization problem.Therefore, the given observations are optimal solutions to this problem for each n, if the objective values of the given observations become upper bounds of $\mathbf{FIOR_n}$. In this case, all optimality gaps are zero. Otherwise, there will be some optimality gaps for some or all observations, denoted as $\mathbf{e}_n^s$. Max(e_n^s) represents the optimality gaps for each user, i.e., the optimal solution for each user is the biggest positive distance, which represents the distance between objective value of the optimal solution and the given solution. The objective function of $\mathbf{MIIO - SUM_1}$ minimizes the summations of the optimality gaps. Constraints (11d) represent the cost coefficient vector for each user n. Constraint (11e) normalizes the attribute values to have a unit measurement for them and (11f) is a normalization constraint that forbids the trivial solution where $\mathbf{f}_n \mathbf{v}_n = \mathbf{0}$. The row vector $\mathbf{f}_n$ represents the item values for the user n and v_n denotes a column vector which shows whether a past user n visited items or not (v_{nm}).

In this problem, there is no assumption of prior knowledge about the unknown matrix. Hence, the matrix obtained from IO is not necessarily close to the real matrix. It is just an

estimate of such a matrix based on the minimization of the summation of optimality gaps. If there is an assumption or prior estimate of the unknown parameters, the objective of the IO model could change to minimize the gap between the obtained and the prior estimate. This idea is similar to the problem solved by Ahuja and Orlin [2001], who assume that a prior estimate of the $\mathbf{c}$ vector is available for IO with a single observation.

Let z_n be the maximum value of $\mathbf{e}_n$ over all extreme points. Therefore, we replace $\max\{e^s\} = z$ and the model (11) will be as follows.

$$\mathbf{MIIO - SUM_2}(\hat{\mathcal{X}}): \quad \underset{a_{pj}, e_n^s, c_n, y_n, z_n}{\text{Minimize}} \quad \sum_{n=1}^{N} z_n \tag{12a}$$

$$\text{s.t.}: \quad (11b), ..., (11f) \tag{12b}$$

$$0 \leq e_n^s \leq z_n \qquad \forall n \in \mathcal{N}, s \in \mathcal{K} \tag{12c}$$

This $\mathbf{MIIO}$ model is useful when the attributes of users and items are identified in advance. Another observation is that the cost vector and the matrix in this model can be negative as well. A negative element in the estimated matrix means that there is a negative relationship between the attributes of users to items. Also, there can be multiple optimal solutions for this problem. Since there is a parameter v_{nm} in the objective function of $\mathbf{FIOR}$, for each variable for which $v_{nm} = 0$, the variable in constraint will be zero, so there is no need to use v_{nm} in the constraints of $\mathbf{FIOR}$.

We can change the objective function of $\mathbf{MIIO - SUM_2}$ to minimizing the maximum of optimality gaps ($\mathbf{MIIO - MAX}$) as follows.

$$\mathbf{MIIO - MAX}(\hat{\mathcal{X}}): \quad \underset{a_{pj}, e_n^s, y_n, z_n, Q}{\text{Minimize}} \quad Q \tag{13a}$$

$$\text{s.t.}: \quad (12b), ..., (12c) \tag{13b}$$

$$z_n \leq Q \qquad \forall n \in \mathcal{N} \tag{13c}$$

We will consider both models $\mathbf{MIIO-SUM}_2$ and $\mathbf{MIIO-MAX}$ in the experimental part to compare the results.

4.3.2.1 Cutting Plane Algorithm for Solving MIIO

Since the models (12a) and (13a) are non-linear, we need to develop an algorithm to solve them. Using a cutting plane algorithm, the goal is to solve an IO problem to find the elements of a matrix $\mathbf{A}$ such that the given multiple data points become optimal or approximately optimal solutions of the forward problem. Algorithm 3 illustrates different steps of implementing the cutting plane algorithm considering model 12a where the IO objective is minimizing summation of optimality gaps. We further refer to it as *Cu-Pl Sum*. This algorithm has similar structure as algorithms 1 and 2 in this thesis. In this algorithm, t denotes number of current iteration.

We first initialize $\mathbb{S}'_n$ which is a subset of $\mathbb{S}_n$, equal to $\emptyset$. Then the algorithm solves an inverse model (11) and finds a matrix of coefficients, i.e., $\mathbf{A}$, and replaces it in the $\mathbf{FIOR}$ model (10) for N different users $n = 1, ..., N$ and solves it for each user. Thus, it obtains the optimal solutions of the forward problems for n users. In each iteration, the algorithm updates the best matrix $\mathbf{A}$ and the best summation of optimality gaps. It then determines whether the given multiple solutions are all optimal for their forward problem. If some of them are not, the algorithm determines how far they are from the optimality and identifies the gaps from optimality. The algorithm terminates as soon as the gaps for all $n = 1, ..., N$ are zero or the summation is equal to the current best summation of errors and the $\mathbf{A}^t = \text{Best } \mathbf{A}$.

Algorithm 3 can also be used for solving model 13a when the IO objective is minimizing the maximum of optimality gaps. For using this model, in line 2 of the algorithm 3, the algorithm solves model $\mathbf{MIIO-MAX}$. It updates the best maximum error in each

Algorithm 3 Cutting Plane Algorithm for Solving $\mathbf{MIIO-SUM}_2$

$t \leftarrow 0$, $\mathbb{S'}_n \leftarrow \emptyset \quad \forall n \in \mathcal{N}$ ▷ Initialization
$\mathbf{A}^0 \leftarrow \mathbf{MIIO-SUM}_2(\hat{\mathcal{X}})$
$\mathbf{x}_n^0 \leftarrow \mathbf{FIOR}(\mathbf{A}^0) \quad \forall n \in \mathcal{N}$
$\epsilon_n^0 = \sum_{m=1}^{M} f_{nm} x_{nm}^0 - \sum_{m=1}^{M} f_{nm} \hat{x}_{nm} \quad \forall n \in \mathcal{N}$
Best $\sum_{n=1}^{N} \epsilon_n \leftarrow \sum_{n=1}^{N} \epsilon_n^0$
$\mathbf{A}^{\text{Best}} \leftarrow \mathbf{A}^0$
while $\{\sum_{m=1}^{M} f_{nm} \hat{x}_{nm} < \sum_{m=1}^{M} f_{nm} x_{nm}^t\} \quad \forall n \in \mathcal{N}$ **do** ▷ Optimality condition
    $\mathbb{S'}_n \leftarrow \mathbb{S'}_n \cup \mathbf{x}_n^t \quad \forall n \in \mathcal{N}$
    $t \leftarrow t + 1$
    $\mathbf{A}^t \leftarrow \mathbf{MIIO}_2(\hat{\mathcal{X}})$
    $\mathbf{x}_n^t \leftarrow \mathbf{FIOR}(\mathbf{A}^t) \quad \forall n \in \mathcal{N}$
    $\sum_{n=1}^{N} \epsilon_n^t = \sum_{m=1}^{M} f_{nm} x_{nm}^t - \sum_{m=1}^{M} f_{nm} \hat{x}_{nm}$
    if $\sum_{n=1}^{N} \epsilon_n^t =$ Best $\sum_{n=1}^{N} \epsilon_n$ **then**
        if $\mathbf{A}^t = \mathbf{A}^{\text{Best}}$ **then**
            Stop
        else
            $\mathbf{A}^{\text{Best}} \leftarrow \mathbf{A}^t$
        end if
    end if
    if $\sum_{n=1}^{N} \epsilon_n^t <$ Best $\sum_{n=1}^{N} \epsilon_n$ **then**
        Best $\sum_{n=1}^{N} \epsilon_n \leftarrow \sum_{n=1}^{N} \epsilon_n^t$
        $\mathbf{A}^{\text{Best}} \leftarrow \mathbf{A}^t$
    end if
end while
Output: $\mathbf{A}^* \leftarrow \mathbf{A}^{\text{Best}} \quad \sum_{n=1}^{N} \epsilon_n^* \leftarrow$ Best $\sum_{n=1}^{N} \epsilon_n$

iteration. The algorithm terminates optimally when all errors are zero or when the maximum error in an iteration is equal to the best maximum error (Max$\epsilon_n^t =$ Best Max ϵ_n) and at the same time current obtained matrix $\mathbf{A}^t$ is equal to Best Matrix$\mathbf{A}$. In this chapter, we will further refer to the cutting plane algorithm for minimizing the maximum of optimality gaps as *CU-Pl Min-Max*.

4.3.3 A Heuristic Approach for Solving IO

The model 10 is the forward integer optimization problem for RS. The LP relaxation heuristic method for minimizing the maximum of optimality gaps is based on solving the following model to optimality:

$$\textbf{HeurM}(\hat{\mathcal{X}}): \quad \underset{\mathbf{c},\epsilon_n^a,\mathbf{y},E}{\text{Minimize}} \quad E \tag{14a}$$

$$\text{s.t.}: \quad y_n \geq f_{nm} v_{nm} \qquad \forall m \in M, n \in \mathcal{N} \tag{14b}$$

$$\sum_{m=1}^{M} f_{nm} v_{nm} \hat{x}_{nm} + \epsilon_n^a = R_n \mathbf{y}_n \qquad \forall n \in \mathcal{N} \tag{14c}$$

$$\epsilon_n^a \leq E \qquad \forall n \in \mathcal{N} \tag{14d}$$

$$f_{nm} = \sum_{p=1}^{P} \sum_{j=1}^{J} u_{np} a_{pj} h_{jm} \quad \forall n \in \mathcal{N} \& m \in \mathcal{M} \tag{14e}$$

$$\mathbf{c}_n = \mathbf{f}_n \mathbf{v}_n \forall n \in \mathcal{N} \tag{14f}$$

$$\sum_{p=1}^{P} \sum_{j=1}^{J} \left| a_{pj} \right| = 1 \tag{14g}$$

$$\left\| \mathbf{c}_n \right\|_1 = 1 \quad \forall n \in \mathcal{N} \tag{14h}$$

In this model, constraint (14b) shows dual feasibility and (14c) is a strong duality constraint. ϵ_n^a represents the differences between the given solutions and optimal solutions for the objective values of the forward LP relaxation problem. The normalization constraints forbid trivial solutions. CPLEX can solve this model without requiring any custom-built algorithm. Solving the inverse problem of the LP relaxation results in a solution that is feasible to the inverse of the original. In this chapter, we further refer to the heuristic approach as *Heur-Sum* if the goal is minimizing the summation of the optimality gaps. Also, we will refer to the heuristic for minimizing the maximum error over all observations as

Heur-Min-Max.

4.4 CASE STUDY: RECOMMENDER SYSTEMS IN RESTAURANTS

As a case study, we consider applying RS in restaurants, which has been considered in the literature, e.g., by Vargas-Govea et al. [2011]. However, the literature studies on restaurant RS and RS in general use machine learning methods. In contrast, in this thesis, we use inverse optimization to recommend the best restaurants to the customers based on the similarities between the attributes.

There are several choices for customers who want to choose restaurants. Each customer has its tastes, and also each restaurant has its specific attributes. Making the best decisions among various choices is challenging. RS is a technique that aims to recommend the best items to the user. In this case, informing about the user attributes is very helpful since the system will know more about the characteristics of each user and can provide a better recommendation.

In the problem of recommending the restaurants to the users, there are m restaurants (items), and the attributes of the restaurants are available as item attributes. Also, there are n customers (users) with their known attributes. Furthermore, the ratings that past users gave to the items are available. The purpose is to use the prior ratings to estimate/impute the user/item attribute weight matrix such that the observed ratings are optimal or close to the optimal to the forward problem. By learning this matrix, we can predict the ratings that any user gives to the items. The users can be the ones that already have some ratings and also for the new users that do not present any item preferences. Based on the user attributes, the item ratings will be recommended to him/her.

4.4.1 Assumption for RS in Restaurants

Based on the available data set obtained from http://chefmoz.org, which will be explained more in details in section 4.4.2, the following assumptions for the model and for the employed data set should be taken into account:

1. Assumptions for data set:

   (a) In the real data set, the rating that users gave to each item was 0 if they did not like the item, 1 if they liked it, and 2 if they liked it very much. Since in **FIOR**, the variables are binary, we pre-process the data by replacing the scores 2 equal to 1. Therefore, for our data set, the score is 0 if a user did not like the restaurant and 1 if he/she liked it. The past users' ratings are the scores that users gave to the visited items.

   (b) In the data set that we use for this work, there are some users who visited some particular restaurants and gave a rating of 1 to all visited restaurants. Also, some users visited some restaurants but gave a score of 0 to all those visited restaurants. We consider the ratings of both of these kinds of users in the prior data set since their decision can affect the results.

   (c) The user attributes are available and obtained explicitly by asking them when they signed up on the website for the first time. Hence, this data is more reliable than observing the users' behavior. We assume that users are optimizing their decisions for rating items.

   (d) The training and test sets are randomly sampled from the data set. There is no manipulation for selecting them, and they are independent of each other. There is no relationship between the users of training and testing. However, if we choose bigger sample sizes, some training users' data can repeat in the sample.

(e) We used the subset of users for whom all the data was available since some information for some items and users were missing in the real data set.

2. Assumptions for model $FIOR_n(R_n)$:

   (a) We assume that all past and new users are in the same city and country. (As future work, we can add a constraint for the forward problem to control the address and location).

   (b) In the objective function of the model, we consider v_{nm}, which shows whether a past user n visited item m or not. For recommending the items to the new users, we assume this vector is 1 for all elements. This way, we let the model choose any item for the user. However, this model can predict ratings for past users with choosing un-visited items. For instance, if there is a user with past ratings for 4 items and weights of the items after obtaining the matrix for him/her are $(-0.2, 0, 0.2, 0.3)$. If the past user could choose 3 items and has already visited item 1,3,4, in this case, the system will recommend items 2,3,4 as an optimal solution instead of 1,3,4. Although the user has not already visited item 2, the solution is better due to the negative value found for item 1.

   (c) There is no prior estimate for the user/item attribute weight matrix.

4.4.2 Data Analysis

We obtained the data set for this study from http://chefmoz.org. These data were also used by Vargas-Govea et al. [2011]. Based on the available information in Vargas-Govea et al. [2011], it is a seven months experiment (i.e., from July 2010 to February 2011). We filtered out any incomplete records. Among the total data set, the finalized number of users for this paper is 116. There are 130 restaurants and 33 attributes for them. Some attributes

belong are related, such as the *smoking area* attribute which can be decomposed into three attributes: *non-smoking area*, *smoking at the bar*, *permitted*. Hence, considering the main attributes, there will be 11 main item attributes: *alcohol area*, *smoking area*, *dress code*, *price*, *accessibility*, *payment*, *parking area*, *franchise*, *ambiance*, *other services like internet*, *open area*. The users' attributes are provided when he/she signs into the system the first time or modify his/her personal information. The main attributes that are considered for users and have been asked them about are: *smoking*, *drink level*, *dress preferences*, *ambiances*, *transportation*, *marital status*, *hijos(kids)*, *interest*, *personality*, *religion*, *activity*, *budget*, *payment methods*. These attributes have been asked with different sub-questions. The user can choose one answer for each question about the attributes. For instance, "Are you: smoker? non-smoker?" "which payment method you prefer? : Debit? Cash? Visa?" and they can choose just one for each question. We consider each sub-element of these attributes as binary data in the data set (smoker (0 or 1), non-smoker (0 or 1)), and if one of them is 1, the other one should be zero. Also, we removed some user attributes, e.g., height, weights, and date of birth. Firstly, a user's height is usually not critical in his/her decision to choose a restaurant. Furthermore, for this case, we just kept those attributes that we could split them into binary values.

We randomly selected instances for doing the experiments among the whole sample. To randomly choose data among 116 users, we used uniform distribution to select samples. So, every single user has the same probability of being chosen. Among 113 users with 45 attributes and 130 items with 33 attributes, we obtained 20 randomly selected instances for each training sample size, including size 5, 10, 15, 20, 25. For selecting data randomly, we gave a random number between 0 and 1 to each user and then sorted them from the smallest to the largest and chose among them after sorting. After sorting once and choosing based on the size of the users, we assigned numbers randomly again and sorted it and re-did the

selection method. Since the total number of available users was 113, for some samples – especially for those with more than 10 users – users can repeat.

The actual ratings of users to the items were 0, 1, 2. However, we pre-processed the data to just consist of 0 and 1 (2 – "liked very much" – was counted as 1, i.e., "liked") because of the limitation of our model (which has binary decision variables). Therefore, the results (ratings) are binary either 0 (dislike) or 1 (like) a restaurant. The rating average for our paper is about 8.49 per user (summation of items that are visited by users/number of users; the total number of users was 116), and the average rating per one user was 6.6. According to the items, the least number of ratings for items was equal to 2 ratings for an item, and the most number of ratings for an item was 30 ratings (30 users assigned a rating of 1 to an item). About 48 items had rating numbers less than 5. Also, 75 of them had ratings from 5 to 20 times, and 7 items had ratings 20 or more times. .

4.4.2.1 Detailed Attributes Information

In Tables 4.1 and 4.2, the detailed attributes utilized for the users and items are presented.

4.4.2.2 Training Sets and Test Sets

Following the paper by Zeng et al. [2016], who considered 20 percent of randomly selected customers as the test set, we take 20 percent of 116 users as a test set and choose them randomly and make them separate from the training set. For the experiments, 23 users out of 116 are in test sets, and the rests are training (93). We do random selection using a uniform distribution for selecting users, meaning that all users have the same probability of being chosen. We choose a random selection method in Excel to choose randomly 23 users. In the training set, there are different sizes of instances, including 5, 10, 15, 20, 25

User Attributes
-*Smoker* (False, True)
-*Drink level* (Abstemious, Social drinker, Casual drinker)
-*Dress preference* (Informal, Formal, No preferences, Elegance)
-*Ambiance* (Family, Friendly, Solitary)
-*Transport* (Foot, Car, Public transport)
-*Marital status* (Single, Married, Widow)
-*Hijos* (Independent, Dependent, Kids)
-*Interest* (Variety, Technology, None, retro, Eco-friendly)
-*Personality* (Thrifty-protector, Hunter-ostentatious, Conformist, Hard-Worker, None)
-*Religion* (None, Catholic, Christian, Mormon, Jewish)
-*Activity* (Student, Professional, Unemployed, Working Class)
-*Budget* (Medium, Low, High)
-*Payment* (Cash Visa, Debit)

Table 4.1: Detailed Names of Attributes for Users

users.

Item Attributes
-*Alcohol* (Non-Alcohol-served, Wine-beer, Full-Bar)
-*Smoking Area* (None, Only at bar, Permitted, Section, Not permitted)
-*Dress code* (Informal, Formal, Casual)
-Accessibility (No, Completely, Partially)
-*Parking lot* (Yes, No, Public Parking)
-Other services (None, Internet, Variety)
-*Area* (Closed, Open)
-*Franchise* (f, t)
-*Ambience* (Familiar, Quiet)
-*Price* (Medium, Low, High)
-*Payment* (Debit, Cash, Visa)

Table 4.2: Detailed Names of Attributes for Items

4.5 EXPERIMENTAL RESULTS

To implement the models and algorithms, the cutting plane algorithm and a heuristic method with minimizing the summation of optimality gaps (*Cu-Pl Sum* & *Heur Sum*) and minimizing the max of optimality gap (*Cu-Pl Min-Max* & *Heur Min-Max*) are considered. We assume that there is no prior information available about the cost vector in advance. After doing experiments, the obtained user/item attribute weight matrix will be used in the forward optimization model to recommend the items to any user. Results illustrate that

Cu-Pl Sum and Cu-Pl Min-Max can find a user/item attribute weight matrix that, when used in the forward model, would result in rating solutions that are close to the actual ones (observed ratings). The results obtained based on the testing set show that our IO method can predict the recommendations well.

The Cu-Pl Sum and Cu-Pl Min-Max algorithms are implemented in a Visual C++ 2013 environment using Concert Technology to call IBM ILOG CPLEX version 12.7.0.0. We run the Heur Sum and Heur min-max using CPLEX.

There are two sets of experiments in this paper: training set experiments and testing set experiments. The goal of the training set experiments is to learn user/item attribute weight matrix given past item ratings. The goal of the testing set experiments is to analyze the accuracy of the item rating predictions for testing users with the utilization of the obtained matrix from training results. They compare the predicted item ratings to the actual ratings. The results of these experiments are shown in Sections 4.5.1 and 4.5.2.

4.5.1 Training Sets to Find the User/Item Attribute Weight Matrix

We consider five different training set sizes: 5, 10, 15, 20, 25 observations. For each size, 20 instances are generated randomly from the data set. Therefore, the total number of instances was 100. The goal is to determine the effect of a different number of observations on the user/item attribute weight matrix. Since the feasible region of $\mathbf{FIOR_n(R_n)}$, for each user is different, having more observations does not necessarily reflect a smaller optimality gap.

In this experiment, we consider "time" as a termination criterion when it is 30 minutes. Table 4.3 reflects the number of problems that solved to optimality in this time frame. As shown in the table, all methods can find an optimal solution for all 20 instances up to size 20 users. Usually, when the number of observations is greater than 20, the algorithms were

Size of Instances	**Cutting Plane Algorithm**		**LP Heuristic**	
Number of Observations	Min-Max	Summation	Min-Max	Summation
5 Users	20	20	20	20
10 Users	20	20	20	20
15 Users	20	20	20	20
20 Users	16	15	19	19
25 Users	1	2	11	11
Total out of 100 instances	77	77	90	90

Table 4.3: Number of Optimal Solutions Found by Different Methods Within 30 Minutes of Run-time

slower because of the complexity of the problem. The results show that for sizes 20 users and 25 users, Heur Sum and Heur Min-mAX can find more optimal solutions in 30 minutes rather than Cu-Pl sum and Cu-Pl Min-Max.

Table 4.4 illustrates the results of Cu-Pl Sum and Heur Sum, while Table 4.5 shows the results of Cu-Pl Min-Max and Heur Min-Max. All numbers round to 2 decimal points. Based on the results, for this data set, the algorithm finds matrices that make the past observations optimal with zero optimality gaps. With changing the set of observations, the results can be different.

The results of Table 4.4 and 4.5 represent that the average of optimality gaps for this data-set for Cu-Pl Sum and Cu-Pl Min-Max are zero. Although it demonstrates that the Heur Sum and Heur Min-Max can not find the matrices that make all observations optimal, the gaps are very small.

Instance Size	**Cutting Plane Algorithm**			**LP Heuristic**		
Number of Users	Avg. # of Iters	Avg. CPU Time (Second)	Avg. Opt Gaps	Avg. CPU Time (Second)	Avg. Opt Gaps	Avg. Gaps Differences
5 users	5.2	4	0	2	0.00	0.00
10 Users	9.35	18	0	4	0.00	0.00
15 Users	15.25	59	0	25	0.00	0.00
20 Users	12.7	336	0	182	0.03	0.03
25 Users**	17.5	682	0	227	0.02	0.02

Table 4.4: Comparing Methods for Minimizing Summation of Gaps

Instance Size	**Cutting Plane Algorithm**			**LP Heuristic**		
Number of Users	Avg. # of Iters	Avg. CPU Time (Second)	Avg. Max Opt Gaps	Avg. CPU Time (Second)	Avg. Max Opt Gaps	Avg. (%) Gap Differences
5 Users	4.6	14	0.00	4	0.01	0.01
10 Users	8.7	40	0	12	0.01	0.001
15 Users	13.65	101	0	30	0.02	0.02
20 Users	12.19	498	0	94	0.08	0.08
25 Users**	13	951	0	349	0.27	0.27

Table 4.5: Comparing Methods for Minimizing Maximum of Gaps

Since we observed that an optimal value of 0 is possible for the given data sets, we conjecture that the users in the data set were similar to each other in terms of their ratings. We expect that for a data set with very different users (i.e., very different attributes), it would not be possible to make all observations optimal, and the optimal value of the inverse problem would not be 0.

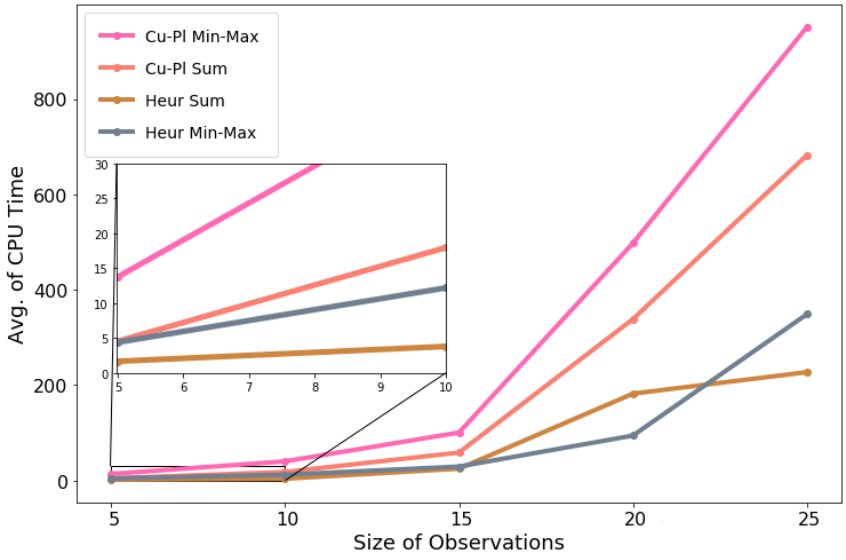


Figure 4.3: Evaluating CPU Time by Increasing Points in Training Sets (seconds)

We proposed the heuristic method to overcome computational issues of the cutting plane algorithm. Figure 4.3 compares the CPU time for different methods. This figure shows that while both cutting plane and heuristic, can solve instances up to the size 10 observations in a short amount of time, the Heur Sum and Heur Min-Max still can solve faster. This is obvious when the size of the observations is 25. However, after size 25, we observed that because of the complexity of the IO problem, none of these methods could work fast.

In contrast to the single point case in Chapter 2, where the forward problem caused computational issues, in this chapter, the issues arise due to the inverse problem, which

cannot be solved quickly. To analyze the sensitivity of the results, we reduced the number of attributes to see whether it can solve faster or not. We observed that by reducing the dimensions of the user/item attribute weight matrix, there was not a considerable change in the computational time. Since Cu-pl Sum and Cu-Pl Min-Max adds n constraints in each iteration, the inverse problem becomes more challenging to solve in each iteration.

Also, we implemented the Heur Sum for two instances obtained from data set with 30 users, and none of them terminated with any optimal solution up to 2 hours. However, with 25 users, the result obtained within 30 minutes. We checked Heur Min-Max to see how it works based on changing the objective. For Heur Min-Max, we tested one sample with 25 users, and it found the solution in 5 hours. The heuristics are implementing slowly since the relating IO model 14 has a lot of constraints and variables. However, for the cutting plane, the algorithm starts initially with a small number of constraints, and it increases in each iteration. Hence, the first iteration is fast and then runs slowly.

4.5.2 Testing Sets for Validating the Obtained Matrix

In this part, we use the user/item attribute weight matrices gathered from the training experiment in Section 4.5.1 as inputs of **FIOR** models for testing. There are 23 testing users. The purpose is to see how good the recommendations will be using the matrices obtained from the training set; that is, how well the fitted forward model can find the ratings of the items for the specific users in the test set. If the obtained matrix imputed from the training experiments, lead to predict the actual scores that past users designated as rating, or the objective values for them are equal, it means that the method works well. This part discusses these results where the optimality gaps address the distinction between objective values of the actual users' ratings and the ratings predicted by solving the forward problem with the imputed matrix. We replace the obtained matrices from training sets (20 matrices

obtained for every 5 different sizes) as parameters in the $\mathbf{FIOR}$ model in the testing set. After attaining optimal solutions for $\mathbf{FIOR_n(R_n)}$, we compare the predicted ratings with the actual ratings that were accessible from the past users.

Size of Instances	**Avg Opt Gaps To the Optimal Decisions of Users**			
Number of Training Users	Cu-Pl Sum	Cu-Pl Min-Max	Heur Sum	Heur Min-Max
5 users	0.08	0.07	0.06	0.18
10 Users	0.05	0.07	0.03	0.08
15 Users	0.05	0.05	0.04	0.05
20 Users	0.11	0.10	0.15	0.13
25 Users	0.02	0.01	0.03	0.14

Table 4.6: Evaluating Prediction Power of Recommender Systems (for 23 Random Test Instances)

Table 4.6 displays the total average of optimality gaps for the 23 test users based on the matrices obtained from various training instances. In these experiments, we solved 23 (test users) * 20 (matrices obtained from training instances) = 460 instances for each size of 5, 10, and 15. Since for sizes 20 and 25 in training, neither the cutting plane algorithm nor the heuristic could find the optimal matrix within 30 minutes for all instances, the total number of obtained matrices are less than 20. Therefore, for these sizes, the total number of solved instances in the testing set is less than 460.

The results of Table 4.6 indicates that all methods work well by increasing the number

of observations up to size 15 since the average of gaps will be reduced. After 15, the results are not comparable with the previous ones since, as already explained, the number of instances is not equal to 460. For this experiment, there is no relation between the training users of different sizes; therefore, we see that for Cu-Pl Sum moving from 10 users to 15, users are not reducing optimality gaps. The same scenario happens for Cu-Pl Min-Max from 5 to 10. In each size, we have a different set of users, and training users are independent. Hence, we do not necessarily obtain better results by increasing the number of observations(training users). Based on using a different number of observations in training sets, we concluded having more observations will not always get a better prediction. It more depends on the observations and their relationships between them.

By observing the results of this table, we conclude that, most of the time, the Cu-Pl Sum and Heur Sum find smaller optimality gaps rather than Cu-Pl Min-Max and Heur Min-Max. Also, a surprising result from Table 4.6 is that Heur Sum works very well, and it can find a smaller gap – even better than Cu-Pl Sum.

Table 4.7 represents the total percentage of solvability of the test sets and the percentage of the number of the sets that can find *optimality gaps* equal to zero.

Table 4.8 represents min, max, and the median average of optimality gaps for 23 users obtained from different matrices in different training sizes.

Size of Instances	**Cu-Pl Sum**		**Cu-Pl Min-Max**		**Heur Sum**		**Heur Min-Max**	
Number of Training Users	Total. % of Optimality	Total. % of Solvability	Total. % of Optimality	Total. % of Solvability	Total. % of Optimality	Total. % of Solvability	Total. % of Optimality	Total. % of Solvability
5 users	71	100	71	100	66	100	40	100
10 Users	65	100	66	100	61	100	50	100
15 Users	60	100	59	100	53	100	52	100
20 Users	53	75	50	85	42	95	43	95
25 Users**	77	7	69	3	52	60	44	75

Table 4.7: Total % Optimal Solutions Using the Matrices Obtained from Training

Size of Instances	**Cu-Pl Sum**			**Cu-Pl Min-Max**			**Heur Sum**			**Heur Min-Max**		
Number of Training Users	Min	Median	Max	Min	Median	Max	Min	Median	Max	Min	Median	Max
5 users	0.02	0.08	0.19	0.02	0.06	0.12	0.01	0.05	0.15	0.02	0.11	0.69
10 Users	0.02	0.05	0.14	0.024	0.07	0.17	0.02	0.03	0.05	0.02	0.05	0.25
15 Users	0.013	0.05	0.09	0.02	0.05	0.09	0.01	0.03	0.15	0.02	0.04	0.14
20 Users	0.02	0.04	0.97	0.03	0.05	0.97	0.02	0.04	0.96	0.02	0.04	1.01
25 Users**	0.01	0.02	0.03	0.014	0.01	0.01	0.01	0.04	0.06	0.02	0.05	0.92

Table 4.8: Min, Max, Median for Optimality Gaps of Testing Result Based on Matrix Obtained from Training

Based on the above results, we observe that sometimes adding an observation(increasing number of training users) to the problem can increase the optimality gap for prediction and sometimes decrease it. From one perspective, by adding more observations to the problem, the prediction should be more accurate since it should be more representative of a broader set of users. However, it appears that the number of observations is not very important. Instead, what is notable is how close the attributes of the observations are to the test user's attributes. We will manifest in the discussion section that if the training users have similar attributes to the testing users, the obtained matrix can lead to a better item ranking prediction. Table 4.9 depicts the percentages of obtaining true positive ratings, and it measures

Size of Instances	**Avg % of Finding Actual Items**			
Number of Traning Users	Cu-Pl Sum	Heur Sum	Cu-Pl Min-Max	Heur Min-Max
5 users	71 %	78 %	69 %	70 %
10 Users	81 %	87 %	80 %	82 %
15 Users	86 %	83 %	85 %	80 %
20 Users	81 %	73 %	81 %	76 %
25 Users	87 %	86 %	87 %	76 %

Table 4.9: Illustrating Percentage of Finding the Actual Items

the proportion of correctly predicted items. This percentage displays the ratio of the total number of *true predicted* item scores to the total number of *actual* scores. The percentages for each size, are quantified as follows: (i) For each obtained matrix from the training set, we calculate the average percentage of finding the actual items for 23 testing users; (ii) According to the results of Table 4.3, up to size 15, 20 instances for each size can be solved and find 20 matrices. The procedure, explained in (i), is repeated for 20 matrices; (iii) The average of the averages attained from (ii) is calculated. The percentages in Table 4.9 shows the accuracy of prediction for all proposed methods. The higher the percentage is (closer to 100 %), the better the algorithms work. Table 4.9 shows that all of these methods can predict the actual ratings up to 70 %.

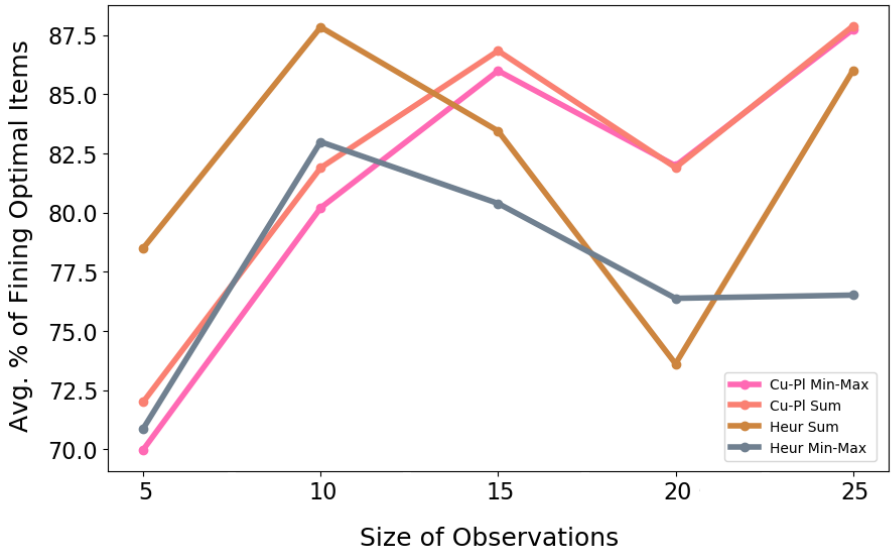


Figure 4.4: Percentage of Finding the Exact Items Using Different Methods

Figure 4.4 demonstrates the percentage of true predictions using various methods. As the line chart shows, all of these methods can predict more than 70 percent of the actual items on average. In size 10, it is noticeable that the Heur Sum can predict actual items up to 85 percent, and also Heur Min-Max works well. While by adding more observations, these

percentages are decreasing dramatically, cutting plane algorithms have growing trends most of the time. The charts illustrate that for size 20, probably the training users are not similar to the testing observations, and they did not help to find a better solution. However, there was progress again in size 25. Recall that for instances of size 20 and 25, the number of instances used to calculate the percentage is not the same as for size 5, 10, and 15 (please see the Table 4.3). Also, there is no relationship between the instances of different sizes, i.e., they are all independent. The results address that the heuristic methods perform well for prediction goals for this problem. Furthermore, based on the experimental training results, the computational times for finding the user/item attribute weight matrix are more efficient than cutting planes. Although, Figure 4.4 displays that for 25 observations the percentage of accuracy prediction by cutting plane is very considerable, generalizing this result needs further investigations.

4.6 DISCUSSION

4.6.1 Quality or Quantity of the Observations?

In this part, we discuss the significance of the similarity between the training and testing sets While one training observation can decrease the optimality gap for the recommendation in the test set, another one can increase it. The purpose is to examine how much the nature of the past observations can change the prediction. We observed that having more observations added to the same data set did not necessarily reflect smaller optimality gaps in the testing set. However, the closer the training users' attributes are to the testing user's attributes, the more the optimality gaps in the test set will decrease. Table 4.10 represents the results of a small example when we randomly add to the observations of the problem. Starting by one observation, we add to the same observation one by one up to 4 observations. In this table, 4* users and 4** users have different attributes. We will analyze

the result of adding user 4* and then user 4**. As the table shows, adding the (4*) to the observations increases the average of optimality gaps for prediction while another observation, 4**, instead of 4*, can decrease the gaps. The reason is that user 4** attributes have similarities with the testing user attributes. This time, it is observable that the optimality gap is decreasing. Therefore, having more data does not guarantee better solutions. As future work, we can cluster the training observations based on the similarities between their attributes and use those users that have more similar attributes. Table 4.10 represents that the results obtained from cutting planes are different from heuristic; by adding more observations, the cutting plane is making the gap smaller, but the gap for heuristic is fluctuating.

Size of Instances	**Avg Opt Gaps To the Optimal Decisions of Users**			
Number of Traning Users	Cu-Pl Sum	Heur Sum	Cu-Pl Min-Max	Heur Min-Max
1 users	0.50	0.50	0.50	0.50
2 Users	0.25	0.22	0.25	0.39
3 Users	0.10	0.10	0.16	0.16
4* Users	0.15	0.17	0.13	0.14
4** Users	0.07	0.28	0.1	0.16

Table 4.10: Sensitivity of Using Different Observations for Testing Results

4.6.2 Reviewing the Effect of the Quality of the Attributes

In this part, we analyze the importance of similarities between the attributes of users. We consider the small size of the training set of users that could have resulted in a good

prediction. Later, we can do further investigation for other sizes to assure that this result can be generalized. In the experimental part, we observed that in an instance of data set with 5 users, the (multi-point) cutting plane algorithm could solve 19 problems out of 23 (in the test set) up to the optimality gaps equal to zero. We examined the attributes of 5 training users to see how much similar they are to the testing users' attributes. Below are the similarities between the attributes of the training and testing users.

Firstly, we compare the similarities between the training users' attributes (observations). The results of this analysis show that among 5 training users (observations), 4 of them have similar attributes. Meaning that 80 percent of the users are non-smoker, single, hijos independent, and prefer to pay by cash. Also, all of the 5 training users (100 percent) have a medium budget and are Catholic. Furthermore, 3 of them (60 percent) are students, hardworking, have their car, and their ambiance is family.

Secondly, we compare the testing user attributes to the training user attributes. Among 23 testing users, 18 users (78 percent) have the same tastes as the training users; They are catholic and prefer to pay cash. 16 users (60 percent) of the test set are non-smokers. About 21 users (90 percent) of the test set are single, and hijos independent, 22 users or 95 percent are students, and most of the personalities are hard workers and thrifty protectors. Therefore, we can conclude that the similarities between user attributes of the training set and the test set will result in a more accurate prediction.

4.7 FUTURE WORK

We suggest several future work directions:

- **Considering the dynamic behavior of users:** The preferences of customers can vary over time. Our framework would need to be adjusted to take this characteristic into account (e.g., by giving more weight to the more recent observations).

- **Customizing the significant user's attributes for each application:** For instance, for application in restaurants, the location of the customer is an important attribute. However, it may not affect choosing an online film. Therefore, in the future, the most effective attributes should be extracted and customized based on the particular case.

- **Determining the most relevant users and removing redundant users based on their attributes:** To have a better solution with smaller optimality gaps, employing the clustering of users for each training instance could be useful. We will cluster based on the similarities in attributes and ratings.

- **Adding budget, location and time, etc. (depending on the subject) constraints to FIOR:** To have more precise results, we should consider constraints that affect the decisions.

- **Comparing the results obtained from IO with other data-driven techniques especially machine learning methods:**To investigate the differences between IO and other data-driven techniques for RS, we should implement the experiments for both methods and analyze the discrepancy between the results.

- **Further analyzing the difference between matrices obtained from the heuristic and cutting plane methods:** To investigate a theoretical reason behind the obtained results arise from the cutting plane algorithm and heuristic method for imputing the user/item attribute weight matrix, we need to focus more on the theoretical analysis.

- **Deciding within a group:** As in the decision-making process such as deciding on dining in a restaurant, sometimes there is not just one decision-maker. It can be a group of family or friends that choose mutually to make a great decision. In this situation, the system requires all members' attributes since it wants to satisfy all

members. As a future direction, we can investigate a method for finding group data ad develop models and algorithms for it.

4.8 CONCLUSION

In this study, we proposed an IO approach with multiple integer observations for an application in RS. Based on past observations, which are the item ratings given by past users, IO imputes the unknown user/item attribute weight matrix to make the ratings optimal or approximately optimal. This matrix can be applied in an optimization problem to predict the ratings and recommend items to the users. We developed a cutting plane algorithm and a heuristic method for solving the IO model. The assumption is that user attributes are available, and also users are optimizing their choices.

We examined a case for employing IO as a technique for recommending restaurants. We split the data into the training and the test sets. Based on the data set, we separated 80% of the data for the training set and the remaining 20% for the test set. Using the training set, we used IO to obtain the user/item attribute weight matrix. We tested these matrices for predicting the ratings. The purpose of the testing was to investigate how well the obtained matrices could predict the same ratings as the actual ratings given by the users. The results of the training illustrated that cutting plane algorithms found matrices that result in zero optimality gap. The heuristic was also able to obtain results that were close to the optimal. Furthermore, the experiments in the test set showed that the matrices imputed using the cutting plane and heuristic algorithms could predict the actual ratings up to an average of 70%.

In comparison with other data-driven techniques, in IO, by taking into account the assumption that the users are optimizing their decisions, we utilize the fewer amount of data. This paper is a preliminary work on using IO for RS. The results have motivated us

for further investigations in the future.

Chapter 5

Conclusions and Future Work

This chapter summarizes this thesis and its main contributions. Then, we propose a few critical directions for future work.

5.1 SUMMARY AND CONTRIBUTIONS

The model parameters of a forward optimization problem are sometimes unknown and hard to estimate. Past solutions to the forward problem can be accessible through historical observations. Given past observations, the goal of IO is to identify the unknown model parameters of the forward problem to make the observations optimal or close to optimal. The assumption is that the observations are coming from an optimization problem, and the experts are optimizing their decisions. Below, we categorize the studies for this dissertation and summarize the contributions and results.

Theoretical Perspective In Chapter 2, we focused on IO with a single, potentially imperfect, integer observation. The observation is assumed to be a feasible solution that may or may not be candidate optimal (i.e., there may not exist a cost vector that makes this solution exactly optimal). The need to consider imperfect observation arises from multiple

sources. First of all, the decision-makers may deviate from deciding optimally. Second, the gathering data process may result in having some erroneous data. To the best of our knowledge, there is no prior work in the literature focusing on IO with an *imperfect integer* observation.

In Chapter 3, we considered *multiple* observations instead of a single observation since having more observations can get a better insight into the problem. In this case, the goal was to identify the unknown coefficients of the objective function to minimize the aggregation of optimality gaps of each given point. We considered *error* for the observations since it is sometimes impossible to simultaneously make all observations optimal. To the best of knowledge, this is the first time that someone considers IO with *multiple imperfect* integer observations.

Finally, we customized the IO approach with multiple imperfect integer observations for Recommender Systems. In a typical recommender system, the unknown parameters are the user/item attribute weights. We assume that user attributes and item attributes and past user ratings of the items are available. Given prior users' ratings, the goal is to determine the unknown user/item attribute weight matrix. By obtaining the values of the elements of this matrix, we can employ it to predict any item rating for any user (future user or previous user that requires a recommendation).

Implementation Perspective In this dissertation, for solving IO models with single and multiple imperfect integer observation(s), new cutting plane algorithms are developed. The currently available cutting plane algorithms in the literature are applicable for a candidate optimal observation, but not in the case of imperfect integer observations. The cutting plane algorithm is efficient for small size problems, but its efficiency declines as the size of the problem and number of observations increase. To overcome the high computational time of the algorithm, we proposed an LP relaxation heuristic method. To the best of our

knowledge, this is the first time that someone considers this heuristic approach for inverse integer optimization problems. The results showed that this heuristic method has better CPU times (as compared to the cutting plane method). Furthermore, although the cutting plane method always results in the smallest optimality error, the differences between the errors obtained from cutting plane and heuristic are relatively low. Therefore, the heuristic can be applied as a replacement of the exact method when CPU time is essential. In Chapter 2, we also proposed the hybrid method that resulted in even a faster algorithm that uses the solution of the heuristic as an initial solution of the cutting plane approach.

Application Perspective To demonstrate a real application of IO with multiple integer observations, we considered a recommender system (RS). RS is a system that can recommend items to users based on past user ratings of items. The system should know the matrix of user/item attribute weights to provide item recommendations. IO is a technique that can identify this unknown matrix using prior ratings. This matrix is an input of an optimization problem that can recommend the best items to any user. There are many data-driven techniques in the literature focusing on RS that utilized a large amount of data for recommending. Given the assumption that the users are optimizing their decisions, IO can use a *small* amount of data for finding an *optimal* recommendation. To the best of our knowledge, no reference in the literature has considered IO for RS. We implemented the cutting plane algorithm and heuristic methods and compared their performance on a real data set for recommending restaurants; specifically, we considered the algorithms' efficiency, differences, and similarities in the sense of prediction, optimization, and computational time. We split the data set into a training set and a testing set. The cutting plane algorithm used the training data set to identify the user/item attribute weight matrices. After imputing this matrix, we used the forward optimization model initialized with this matrix to predict the ratings of the users in the test set. We found that our IO approach could predict the correct

item ratings for more than 70 percent of test instances. Therefore, we demonstrated the use of IO as a method for prediction.

5.2 FUTURE WORK

In this section, we summarize a few directions for future work based on the outcomes of this dissertation.

5.2.1 Theoretical Directions

Based on the theoretical studies developed in this thesis, we recommend the following future work:

- **Considering *uncertainty* in decision-making:** According to Aggarwal and Philip [2008], many current advanced technologies can record large quantities of data. In some cases, this data may contain errors or may be incomplete. This issue has created a requirement for developing algorithms for managing uncertain data. The uncertainty can arise from the inaccuracy of the observed numbers due to various reasons, e.g., the error due to mis-measurement or not being able to observe the true value (e.g., in a stochastic setting). A paper by Ghobadi et al. [2018] focused on uncertainty in the behavior of decision-makers. In this paper, instead of studying multiple points, they assumed a decision maker's behavior as a set of all possible realization of input data. They developed an IO model that finds a cost vector that minimizes the worst-case fit error associated with a realization of an uncertain input data point. While Ghobadi et al. [2018] addressed uncertainty in the decision maker's behavior, we suggest another kind of uncertainty when the given solutions are "interval" numbers. The idea of interval analysis arises from Moore [1966], which states that almost any scientific computation begins with inexact initial data. Interval arithmetic

provides an ideal solution for incorporating measurement uncertainties directly into a calculation. For obtaining past solutions, the system can ask the decision-makers to fill in a survey about their ratings and provide some scores to the items. Sometimes, it is hard for them to assign an exact value for items. Instead, they consider a lower bound and an upper bound for showing their scores. Developing IO models that can deal with such uncertainties is another future direction.

- **Developing IO models considering *time-series* with *multiple observations:*** Assume that we have a sequence of decisions observed at different points in time, i.e., decisions $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$ that can be viewed as a time series but where each decision $\mathbf{x}_t$ is a solution to an optimization problem solved at time t. The goal of IO is to learn a cost vector using these multiple, time-dependent observations. Then, the cost vector can be applied to predict the actions for the following periods. A few references have focused on forecasting using IO, such as Amin and Emrouznejad [2007] and Saez-Gallego and Morales [2017]. What distinguishes our proposed future work from others is that the purpose is to work on an IO model where the exact time-series function is available. To extend this, we recommend to study cases where the precise distribution of time-series is not available, and only time-dependency between observations and their optimality is known. We can further extend such studies to include integer observations.

- **Considering infeasible observations as prior data:** The available data can be imperfect due to the data collection errors or a deviation in decision making. In this thesis, we assumed that past observations are feasible for the forward problem. We can later consider infeasible integer observations to make IO models more practical.

- **Adding *budget, location, and time* constraints to the forward problem, especially**

for RS: To have more precise results, we should consider essential constraints that affect a user's decision. For instance, in choosing the best restaurants for a user, budget, location, and time of day may impose constraints on the user's choices of restaurants. Therefore, these constraints can be considered in the forward problem.

5.2.2 Implementation Directions

To improve the results of the proposed IO models, we propose the following future work to get accurate results in a reasonable computational time:

- **Adding a heuristic to the cutting plane algorithm:** To avoid adding repetitive cuts, we should add a heuristic to the algorithm. Based on the current IO model and the cutting plane algorithm developed in this thesis, the obtained extreme point is likely to be known from the previous iterations. Hence, it is essential not to let the algorithm add duplicate constraints to the IO problem. This way, we can improve the computational time as well.

- **Developing alternative algorithms:** High computational time and the high number of iterations for large scale problems are important limitations of using the cutting plane algorithm. This work was the first attempt for IO with imperfect integer observation(s). Although we have proposed a heuristic approach to deal with these limitations, it may be possible to develop other, more efficient methods.

5.2.3 Application Directions

Future work based on the application of IO in RS is summarized as follows:

- **Considering dynamic behavior of users:** A user's preferences and attributes can change over time. Developing methods to take these changes into account is essential.

- **Selecting the most important attributes for each case:** In future work, for each particular application, we should use methods to extract the most important attributes and customize them based on each particular case. This idea comes from the feature selection technique Vargas-Govea et al. [2011]. This way, we can reduce the dimensions of the user/item attribute weight matrix.

- **Clustering users in data sets and considering more relevant clusters:** The idea of clustering in data is taking clustering algorithms to partitions the users having similar preferences and then apply the recommender systems algorithm to each cluster Das et al. [2014]. This way, they tried to reduce the computational time of the RS algorithm by removing redundant users that can produce irrelevant information. We can cluster the users in the data sets before manipulating them and utilizing them to have a more relevant solution in a shorter amount of time.

- **Comparing IO with machine learning techniques for RS:** We should compare the results obtained from IO with other data-driven techniques, especially machine learning methods for RS to analyze their differences, strengths, and weaknesses for this particular application.

- **Deciding within a group:** In decision-making processes such as deciding on dining in a restaurant, sometimes you are not the only one who votes. You are a group member of family or friends who needs to decide where to eat together. In this situation, the system requires all members' attributes. Also, the recommendation should consider the satisfaction of all members. Typically, the data for group decision making in recommender systems are not easily accessible mainly due to the design of customer surveys. For example, one possible solution is to ask the users who enter the system to clarify whether they are individuals or deciding within a group.

- **Considering Other Applications:** Applying the IO problem with multiple integer observations in other applications can be an interesting future direction.

- **Considering different weights for the error coming from different observations:** In this study, we assume a unit weight for all optimality errors. In future work, we can consider different weights for different observations.

In this thesis, we first reviewed the IO literature based on different categories. In Chapter 2, we developed mathematical models for IO with an *imperfect integer observation*. We then proposed a cutting plane algorithm, an LP relaxation heuristic, and a hybrid algorithm for this problem. We generated random problem instances for testing the performance of our proposed methods. Based on the results of our experiments, we concluded that a cutting plane algorithm is an efficient method for problems with a small set of observations. By applying heuristic and hybrid methods, we decreased the CPU time. We also discovered that heuristic methods could get results that are very close to the optimal. Therefore, whenever the optimality of the solution is not essential, the heuristic method can be an alternative.

In Chapter 3, we generalized IO for *multiple imperfect integer observations*. After developing the mathematical models and implementing the cutting plane and heuristic algorithms, we compared the results of a cutting plane algorithm based on multiple observations and a single observation. We demonstrated that by accessing more observations, we could find the unknown parameters which are closer to the optimal global solution. Hence, when all given observations come from the same feasible regions, it is better to have multiple observations instead of a single one.

In Chapter 4, we proposed an application in recommender systems and employed real data for recommending restaurants to the users using IO. We updated the IO model with a multi-point in chapter 3 for applying in RS. In contrast to Chapter 3, in this chapter, there are multiple feasible regions (a different feasible region for every user). We applied

IO to impute the matrix of user/item attribute weights. Knowing this matrix will help the system to predict the item ratings and recommend the best items to a future user. Assuming that users are optimizing their decisions, we need less prior data for the IO approach as compared to the other methods. We employed real data sets for recommending restaurants to the users. We partitioned the data into training and test sets. We proposed a cutting plane algorithm and a heuristic to implement the models for RS. Based on the experimental results in this chapter, IO could obtain the user/item attribute weight matrices from the testing set that was able to predict the actual data in the test set with an accuracy percentage of over 70 percent.

In Chapter 5, we summarized this dissertation and proposed several future work directions. This dissertation has contributed to the development of algorithms for inverse integer programming with imperfect data and demonstrated a novel application of inverse integer programming.

BIBLIOGRAPHY

Charu C Aggarwal and S Yu Philip. A survey of uncertain data algorithms and applications. *IEEE Transactions on Knowledge and Data Engineering*, 21(5):609–623, 2008.

Ravindra K Ahuja and James B Orlin. Inverse optimization, part i: linear programming and general problems. *Sloan School of Management, Cambridge, MA*, 1998.

Ravindra K Ahuja and James B Orlin. Inverse optimization. *Operations Research*, 49(5): 771–783, 2001.

Gholam R Amin and Ali Emrouznejad. Inverse forecasting: A new approach for predictive modeling. *Computers & Industrial Engineering*, 53(3):491–498, 2007.

Asim Ansari, Skander Essegaier, and Rajeev Kohli. Internet recommendation systems, 2000.

A. Aswani, Z.-J. Shen, and A. Siddiq. Inverse optimization with noisy data. *Operations Research*, 66(3):870–892, 2018.

Aaron Babier, Timothy C. Y. Chan, Taewoo Lee, Rafid Mahmood, and Daria Terekhov. A unified framework for model fitting and evaluation in inverse linear optimization. *arXiv preprint arXiv:1804.04576*, 2019.

Linas Baltrunas and Xavier Amatriain. Towards time-dependant recommendation based on implicit feedback. In *Workshop on context-aware recommender systems (CARS'09)*, 2009.

Dimitris Bertsimas, Vishal Gupta, and Ioannis Ch Paschalidis. Data-driven estimation in equilibrium using inverse optimization. *Mathematical Programming*, 153(2):595–633, 2015.

Stephanie Blanda. Online recommender systems–how does a website know what i want? *American Mathematical Society. Retrieved October*, 31, 2016.

Justin J Boutilier, Taewoo Lee, Tim Craig, Michael B Sharpe, and Timothy CY Chan. Models for predicting objective function weights in prostate cancer imrt. *Medical physics*, 42 (4):1586–1595, 2015.

Aykut Bulut and Ted K Ralphs. On the complexity of inverse mixed integer linear optimization. Technical report, COR@ L Laboratory Report 15T-001-R3, Lehigh University, 2016.

Rainer E Burkard and Ulrich Pferschy. The inverse-parametric knapsack problem. *European journal of operational research*, 83(2):376–393, 1995.

D. Burton and Ph. L. Toint. On an instance of the inverse shortest paths problem. *Mathematical Programming*, 53(1-3):45–61, 1992a.

D. Burton and Ph. L. Toint. On the use of an inverse shortest paths algorithm for recovering linearly correlated costs. *Mathematical Programming*, 63(1-3):1–22, 1994.

Didier Burton and Ph L Toint. On an instance of the inverse shortest paths problem. *Mathematical Programming*, 53(1):45–61, 1992b.

T. C. Y. Chan, T. Lee, and D. Terekhov. Inverse optimization: Closed-form solutions, geometry, and goodness of fit. *Management Science*, 65(3):1115–1135, 2019.

Timothy CY Chan and Neal Kaw. Inverse optimization for the recovery of constraint parameters. *European Journal of Operational Research*, 2019.

Timothy CY Chan, Tim Craig, Taewoo Lee, and Michael B Sharpe. Generalized inverse multiobjective optimization with application to cancer therapy. *Operations Research*, 62 (3):680–695, 2014.

Hung-Hsuan Chen, Liang Gou, Xiaolong Zhang, and Clyde Lee Giles. Collabseer: a search engine for collaboration discovery. In *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*, pages 231–240. ACM, 2011.

Hung-Hsuan Chen, II Ororbia, G Alexander, and C Lee Giles. Expertseer: a keyphrase based expert recommender for digital libraries. *arXiv preprint arXiv:1511.02058*, 2015.

Joseph YJ Chow and Will W Recker. Inverse optimization with endogenous arrival time constraints to calibrate the household activity pattern problem. *Transportation Research Part B: Methodological*, 46(3):463–479, 2012.

Joydeep Das, Partha Mukherjee, Subhashis Majumder, and Prosenjit Gupta. Clustering-based recommender system using principles of voting theory. In *2014 International Conference on Contemporary Computing and Informatics (IC3I)*, pages 230–235. IEEE, 2014.

Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

Zhaoyang Duan. *Parallel cutting plane algorithms for inverse mixed integer linear programming*. Iowa State University, 2009.

Michael D Ekstrand, John T Riedl, Joseph A Konstan, et al. Collaborative filtering recommender systems. *Foundations and Trends® in Human–Computer Interaction*, 4(2): 81–173, 2011.

Zeynep Erkin, Matthew D Bailey, Lisa M Maillart, Andrew J Schaefer, and Mark S Roberts. Eliciting patients' revealed preferences: an inverse markov decision process approach. *Decision Analysis*, 7(4):358–365, 2010.

P. M. Esfahani, S. Shafieezadeh-Abadeh, G. A. Hanasusanto, and D. Kuhn. Data-driven inverse optimization with imperfect information. *Mathematical Programming*, 167(1): 191–234, 2018.

András Faragó, Áron Szentesi, and Balázs Szviatovszki. Inverse optimization in high-speed networks. *Discrete Applied Mathematics*, 129(1):83–98, 2003.

Alexander Felfernig, Klaus Isak, Kalman Szabo, and Peter Zachar. The vita financial services sales support environment. In *Proceedings of the National Conference on Artificial Intelligence*, page 1692. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

C. Gerhards, S. Pereverzyev, and P. Tkachenko. A parameter choice strategy for the inversion of multiple observations. *Advances in Computational Mathematics*, 43(1):101–112, 2017.

Kimia Ghobadi, Taewoo Lee, Houra Mahmoudzadeh, and Daria Terekhov. Robust inverse optimization. *Operations Research Letters*, 46(3):339–344, 2018.

Ali Goli. *Sensitivity and Stability Analysis for Inverse Optimization with Applications in Intensity-Modulated Radiation Therapy*. PhD thesis, University of Toronto, 2015.

Siming Huang. Inverse problems of some np-complete problems. In *International Conference on Algorithmic Applications in Management*, pages 422–426. Springer, 2005.

Yuanchun Jiang, Jennifer Shang, and Yezheng Liu. Maximizing customer satisfaction through an online recommendation system: A novel associative classification model. *Decision Support Systems*, 48(3):470–479, 2010.

Arezou Keshavarz, Yang Wang, and Stephen Boyd. Imputing a convex objective function. In *Intelligent Control (ISIC), 2011 IEEE International Symposium on*, pages 613–619. IEEE, 2011.

Taewoo Lee, Muhannad Hammad, Timothy CY Chan, Tim Craig, and Michael B Sharpe. Predicting objective function weights from patient anatomy in prostate imrt treatment planning. *Medical physics*, 40(12), 2013.

SS Li, Peter Brucker, Chi To Ng, TC Edwin Cheng, Natalia V Shakhlevich, and JJ Yuan. A note on reverse scheduling with maximum lateness objective. *Journal of Scheduling*, 16(4):417–422, 2013.

miplib2017. MIPLIB 2017, 2018. http://miplib.zib.de.

M Moghaddass and D Terekhov. Inverse integer optimization with a noisy observation, 2019. Submitted to Operations Research Letters.

Ramon E Moore. *Interval analysis*, volume 4. Prentice-Hall Englewood Cliffs, NJ, 1966.

University of Florida. Useful Pharmacokinetic Equations, 2000. ISSN 1940-6029. URL http://pharmacy.ufl.edu/files/2013/01/5127-28-equations.pdf.

QHull Library. scipy.spatial.ConvexHull, 2018. https://docs.scipy.org/doc/scipy-0.19.0/reference/generated/scipy.spatial.ConvexHull.html.

R Core Team. R: A Language and Environment for Statistical Computing, 2017. URL https://www.r-project.org/.

Balasubramanian Ram and Sanjiv Sarin. An algorithm for the 0-1 equality knapsack problem. *Journal of the Operational Research Society*, 39(11):1045–1049, 1988.

Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

Julien Roland, José Rui Figueira, and Yves De Smet. The inverse $\{0, 1\}$-knapsack problem: theory, algorithms and computational experiments. *Discrete Optimization*, 10(2):181–192, 2013.

Javier Saez-Gallego and Juan Miguel Morales. Short-term forecasting of price-responsive loads using inverse optimization. *IEEE Transactions on Smart Grid*, 2017.

Andrew J Schaefer. Inverse integer programming. *Optimization Letters*, 3(4):483–489, 2009.

Charlie Sharpsteen and Cameron Bracken. *tikzDevice: R Graphics Output in LaTeX Format*, 2016. URL https://cran.r-project.org/package=tikzDevice.

Carson Sievert, Chris Parmer, Toby Hocking, Scott Chamberlain, Karthik Ram, Marianne Corvellec, and Pedro Despouy. *plotly: Create Interactive Web Graphics via 'plotly.js'*, 2017. URL https://cran.r-project.org/package=plotly.

Steven S. Skiena. Knapsack Problem, 1997. https://www8.cs.umu.se/kurser/TDBA77/VT06/algorithms/BOOK/BOOK4/NODE145.HTM.

A. Tarantola. *Inverse problem theory and methods for model parameter estimation*, volume 89. SIAM, 2005.

Albert Tarantola. Inverse problem theory: Method for data fitting and model parameter estimation. *Elsevier*, 613, 1987.

Marvin D Troutt. A maximum decisional efficiency estimation principle. *Management Science*, 41(1):76–82, 1995.

Marvin D Troutt, Suresh K Tadisina, Changsoo Sohn, and Alan A Brandyberry. Linear programming system identification. *European journal of operational research*, 161(3): 663–672, 2005.

Marvin D Troutt, Wan-Kai Pang, and Shui-Hung Hou. Behavioral estimation of mathematical programming objective function coefficients. *Management science*, 52(3):422–434, 2006.

Marvin D Troutt, Alan A Brandyberry, Changsoo Sohn, and Suresh K Tadisina. Linear programming system identification: The general nonnegative parameters case. *European Journal of Operational Research*, 185(1):63–75, 2008.

Sarina DO Turner and Timothy CY Chan. Examining the LEED rating system using inverse optimization. *Journal of Solar Energy Engineering*, 135(4):040901, 2013.

Blanca Vargas-Govea, Gabriel González-Serna, and Rafael Ponce-Medellın. Effects of relevant contextual features in the performance of a restaurant recommender system. *ACM RecSys*, 11(592):56, 2011.

Lizhi Wang. Cutting plane algorithms for the inverse mixed integer linear programming problem. *Operations Research Letters*, 37(2):114–116, 2009.

Hadley Wickham. Reshaping data with the reshape package. *J Stat Softw*, 21(12), 2007. URL http://www.jstatsoft.org/v21/i12/paper.

Jun Zeng, Feng Li, Haiyang Liu, Junhao Wen, and Sachio Hirokawa. A restaurant recommender system based on user preference and location in mobile environment. In *2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, pages 55–60. IEEE, 2016.

Feng Zhang, CT Ng, G Tang, TCE Cheng, and YHV Lun. Inverse scheduling: applications in shipping. *International Journal of Shipping and Transport Logistics*, 3(3):312–322, 2011.

Jianzhong Zhang and Zhenhong Liu. Calculating some inverse linear programming problems. *Journal of Computational and Applied Mathematics*, 72(2):261–273, 1996.