Prescriptive analytics in operations problems: a tree ensemble approach

by

Max Biggs

BE(Hons), University of Auckland (2013)

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY IN OPERATIONS RESEARCH

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2019

Author .....
Sloan School of Management
29 July, 2019

Certified by .....
Georgia Perakis
William F. Pounds Professor of Management
Thesis Supervisor

Accepted by .....
Patrick Jaillet
Dugald C. Jackson Professor,
Co-director, Operations Research Center

Prescriptive analytics in operations problems: a tree ensemble approach

by

Max Biggs

Submitted to the Sloan School of Management
on 29 July, 2019, in partial fulfillment of the
requirements for the degree of
DOCTOR OF PHILOSOPHY IN OPERATIONS RESEARCH

Abstract

The main contributions of this thesis concern addressing challenges in the field of prescriptive optimization, and how machine learning techniques can be incorporated into solving data-driven operational optimization problems.

In chapter 2, we provide a data-driven study of the secondary ticket market. In particular we are primarily concerned with accurately estimating price sensitivity for listed tickets. We propose a semi-parametric model for measuring heterogeneous treatment effects using the concept of orthogonalization in the classification setting, and derive a novel loss function which can be solved using a range of off-the-shelf machine learning methods. Over a wide range of synthetic data experiments, we show how this approach beats state-of-the-art machine learning and causal inference methods for estimating treatment effects in classification tasks.

In chapter 3, we show how to solve optimization problems with random forest objective functions and general polyhedral constraints. We show how to formulate this problem using MIO techniques and show this formulation can be decomposed and solved iteratively using Pareto-optimal Benders cuts. We also provide analytical guarantees on an approach that approximates a large scale random forest optimization problem by optimizing over a smaller forest, and develop heuristics based on ideas from cross validation.

In chapter 4, we study a new problem where nurse practitioners need to be dynamically routed to patients' houses as service requests are received. We show how to solve using Approximate Dynamic Programming and develop methods to solve ADP's with combinatorial action spaces and non-linear cost-to-go functions approximated using a tree or tree ensemble approximation.

In chapter 5, we propose a Markov Decision Process (MDP) model for the tramp shipping that captures the dynamic and stochastic nature of spot cargo availability. We propose a novel methodology for solving this MDP in a tractable way, by introducing a ranking algorithm which is equivalent to solving the DP. We show that our ranking algorithm outperforms several benchmarks and the average performance of

ships operating on the spot market in practice by between 4% and 32%.

Thesis Supervisor: Georgia Perakis
Title: William F. Pounds Professor of Management

Acknowledgments

First and foremost I would like to sincerely thank my advisor Professor Georgia Perakis. Thank you for being an incredibly kind and caring mentor, who has always had my best interests at heart and was willing to go the extra mile to ensure I had the best opportunities possible, both during my PhD and beyond. Thanks for giving me the freedom to explore ideas I am passionate about while also keeping me focused on the things that needed to get done. Thanks for being a great role model as a professor; someone I will always look up to and hope to continue to learn from.

Thanks to my Thesis Committee — Prof. Nagesh Gavirneni, Prof. Nikos Trichakis and Prof. Tauhid Zaman for their invaluable advice about my thesis and guidance on careers in academia. Thanks also to Nagesh Gavirneni for the opportunity to be your teaching assistant and giving me a great example of how to be an effective teacher.

I would like to thank my research collaborators. First, thank you to my good friend Rim Hariss who has collaborated with me closely on chapters 2 and 3. Thanks for the lengthy discussions and helpful advice on a range of professional and not-so-professional topics. You really brought out the best in me, bringing rigor to my often whimsical ideas. I am very proud of our research projects and very much enjoyed our time working together. Thank you also to Michael Li and Charles Herrmann for being fantastic Master of Business Analytics Students, with research skills far beyond their years. You both have very bright futures ahead of you.

I would like to extend my sincere thanks also to the many industry collaborators I have worked with. Thanks to Maria Kartalou, Christine Roth, Themistoklis Stefanakis and Georgios Zonzilos for your expertise, for your generous hospitality and for making my stays in Vouliagmeni so enjoyable. Thank you to Michael Alley for your willingness to embrace new ideas and analytics in the ticket resale market. Finally, thanks to Michael Elliot, Eric Parks, and Carlos Estrada for your bold ideas and passion for transforming the home-healthcare space. I hope the start-up continues to have success and that one day I will be able to request a nurse practitioner to serve me in the comfort of my own home! Thank you all for teaching me about your respective

industries and for the guidance on how to make my research relevant to practitioners. I am very grateful for the new ideas that have resulted from these collaborations.

I would also like to thank my UROPs, Ali Soylemezoglu, Sarah Caso, Stephanie Hu, Danielle Fang for all your hard work and for developing much of the software that underlies the applications in chapters 4 and 5. I wish you all the best for the future and I'm sure you will all do great things. Thanks also to Georgia's other students Lennart Baardman, Tamar Cohen, Divya Singvi, Ioannis Spantidakis, Omar Skali Lami, Daniel Chen, Charles Thraves, Elisabeth Paulson, Anna Papush for friendship, very useful research feedback and presentation advice.

I am also indebted to those in the Department of Engineering Science at the University of Auckland for introducing me to Operations Research, as well as Giles Moiser for laying the mathematical foundations I have needed to succeed.

Special thanks to the rotating cast of 238 Prospect Street — Greg Monohan, Andy Kriebel, Lieutenant Grant Genzman, Hugh Carson, Spencer Wyant, Max Burq, Sebastien Martin, with special guests Annaëlle Pema and Alena Spielmann. You all made Cambridge a home away from home and I am very grateful for our time together. Thanks for discussions on topics as wide ranging as the merits of sheep-based trading strategies, the origin of the whale noises, what happened to all the whipped cream, and for establishing just how loud it is physically possible to listen to the Soviet national anthem. Thanks also all the friends I have made through the ORC and wider community: Andrew Li, Elisabeth Paulson, Joey Huchette, Zach Saunders, Andrew Vanden Berg, Mathieu Dahan, Anna Papush, Virgile Galle, Emily Meigs, Zach Owen, Charles Thraves, Arthur Flajolet, Cécile Casses, Ludovica Rizzo, Mariapaola Testa, Krishnan Rajagopalan, Colin Pawlowski and everyone else — there are far too many to mention here. Thanks for the many (mis)adventures, high-stakes games, mysteries solved and in general making the last 5 years the some of the best of my life. Thanks to the crew of the HMS Nemo for your bravery at sea and for rescuing HMS Maverick from almost certain disaster several times. Thanks also to Parker Vascik for being a truly capable first mate and embodying the best qualities thereof. Thanks to the Kiwi community at MIT, providing me with friendship and

pies at opportune moments to prevent me from missing home too much. Thanks to the ORC soccer team, both the mainstays and everyone else who played for us over years. In particular, thanks to Alexander Remorov and Joey Huchette for their spirited captaincy and organization. Thanks also to Andrew Vanden Berg for almost singlehandedly dragging us to the finals of the prestigious intramural competition. Thanks also to the Harvard Physics community for adopting me as one of their own, even though I am so clearly not.

Special thanks to Paloma Ocola, for always brightening my mood even in the depths of proof-not-working-deadline-tommorow hell and making the second half of my PhD immeasurably better than the first. Finally, thanks to my family for being my biggest cheerleaders and for always supporting me unconditionally in everything I do, for which I am forever grateful.

Contents

1 Introduction **19**

1.1 Contributions 24

2 Pricing for heterogeneous products: analytics for ticket reselling **29**

2.1 Introduction 29

2.2 Literature review 33

2.2.1 Causal inference 33

2.2.2 Ticket pricing for sporting events 35

2.2.3 Data-driven pricing 35

2.3 Heterogeneous treatment effect estimation for classification 36

2.3.1 Method 38

2.3.2 Consistency and stability of estimator $\hat{\tau}(\cdot)$ 42

2.4 Testing the method: experiments on synthetic data 44

2.4.1 Homogeneous treatment effect simulations 44

2.4.2 Heterogeneous treatment effect: gradient boosted trees 45

2.5 Arbitrage opportunity identification 50

2.5.1 Static pricing 50

2.5.2 Time-varying pricing 52

2.5.3 Global optimization for purchasing a ticket portfolio 54

2.5.4 Business considerations 55

2.6 Implementation 56

2.6.1 Feature generation 56

2.6.2 Instrumental variables 56

2.6.3 Predictive accuracy . . . 58
2.6.4 Probability calibration . . . 59
2.6.5 Back-testing optimization . . . 61
2.7 Conclusions . . . 62

3 Optimizing objective functions determined from random forests 65

3.1 Introduction . . . 65
3.2 Relevant literature . . . 70
3.3 Outline . . . 73
3.4 Model . . . 73
3.4.1 MIP Formulation . . . 75
3.5 Benders decomposition . . . 76
3.6 Approximating large random forests . . . 79
3.6.1 Cross-validation for optimization . . . 88
3.7 Case study: property investment . . . 90
3.7.1 Data . . . 90
3.7.2 Prediction . . . 91
3.7.3 Formulation . . . 91
3.8 Jury selection case study . . . 94
3.8.1 Background . . . 94
3.8.2 Predicting jury verdict . . . 94
3.8.3 Jury selection formulation . . . 96
3.9 Numerical experiments . . . 97
3.9.1 Testing environment . . . 97
3.9.2 Solution time . . . 98
3.9.3 Cross-validation experiments . . . 99
3.10 Comparison with optimizing over other machine learning objective functions . . . 100
3.10.1 Testing environment . . . 100
3.10.2 Results . . . 102

3.10.3 Synthetic data . . . 104
3.11 Conclusions . . . 105

4 Dynamic routing with tree based value function approximations 107
4.1 Introduction . . . 107
4.2 Relevant literature . . . 110
4.2.1 Dynamic vehicle routing . . . 110
4.2.2 Approximate dynamic programming . . . 111
4.2.3 Home health care scheduling . . . 111
4.3 Contributions . . . 112
4.4 Model . . . 113
4.5 Approximate methods . . . 117
4.5.1 General formulation . . . 121
4.5.2 Myopic LP . . . 122
4.6 Mixed Integer Optimization formulations . . . 122
4.6.1 MIO formulation for a single tree . . . 123
4.6.2 Ideal formulation for a single tree with binary features . . . 129
4.6.3 Combined tree and assignment polytope . . . 131
4.6.4 Extensions to tree ensembles . . . 133
4.6.5 Enumerate then optimize . . . 134
4.7 Numerical experiments . . . 135
4.7.1 Solution time . . . 135
4.7.2 Performance of approximate dynamic programming approaches 139
4.7.3 Insights . . . 143
4.8 Extensions to other operations management problems . . . 145
4.8.1 2-way online matching problem . . . 147
4.8.2 3-way online matching problem . . . 149
4.9 Conclusions . . . 150

5 A ranking algorithm for tramp shipping in the spot market 153
5.1 Introduction . . . 153

5.1.1 Problem description . . . 154
5.1.2 Relevant Literature . . . 155
5.1.3 Contributions . . . 158
5.2 Deterministic shipping rate model . . . 160
5.3 Ranking algorithm . . . 162
5.3.1 Ranking algorithm over polymatroids . . . 163
5.3.2 The equivalence between Bellman's equation and the polymatroid LP . . . 165
5.4 Infinite horizon . . . 168
5.4.1 Rank and iterate . . . 169
5.4.2 Infinite horizon ranking LP . . . 171
5.5 Perfect information benchmark . . . 172
5.6 Approximate benchmark methods . . . 174
5.6.1 Numerical results . . . 177
5.6.2 Simulation results . . . 178
5.7 Uncertainty in shipping rates . . . 182
5.7.1 Independent shipping rates . . . 183
5.8 Conclusions . . . 187

6 Conclusions **189**

A Appendix for pricing for heterogeneous products: analytics for ticket reselling **191**

A.1 Derivation of loss function with endogeneity . . . 191
A.2 Proof of theorem 1 . . . 193
A.3 Proof of optimal price strategy . . . 197
A.4 Feature importance and few feature formulation . . . 199
A.4.1 Feature importance . . . 199
A.4.2 Few feature formulation . . . 200
A.5 Correlation between tickets . . . 201
A.5.1 Technical details for covariance test . . . 202

A.6 Additional numerical experiments . . . 203
A.6.1 Treatment effect error with sample size . . . 203
A.6.2 Optimal treatment prescription simulations . . . 204
A.7 Backtesting framework . . . 205

B Appendix for optimizing objective functions determined from random forests 209

B.1 Benders subproblem results . . . 209
B.1.1 Calculation of optimal dual solution . . . 209
B.1.2 Benders Pareto-optimal cuts . . . 211
B.2 Hoeffding difference proof . . . 213
B.3 Bound of suboptimality in the size of M . . . 214
B.4 Hazard rate dominance numerical experiments . . . 215
B.5 Proof of bounds for cross validation . . . 216
B.6 Optimizing over a logistic regression objective . . . 217
B.7 Cross-validation . . . 218
B.7.1 Cross-validation, tree size and overfitting . . . 218
B.8 Jury behavior . . . 220

C Appendix for a ranking algorithm for tramp shipping in the spot market 221

C.1 Data and inference . . . 221
C.2 Dynamic programming example . . . 222
C.3 Ranking algorithm example . . . 225

List of Figures

2-1 Homogeneous treatment effect estimate accuracy ($n = 1000$) 46
2-2 Estimating treatment effect for changing sample size 48
2-3 Estimating change in probability with a change in treatment 49
2-4 Sensitivity of optimal price and revenue to magnitude of treatment effect 52
2-5 Performance on NBA data with IV: ROC curve 59
2-6 Calibration curves on NBA data 60

3-1 Example tree for classifying gender 68
3-2 Example tree to illustrate formulation with given x belonging to leaf 6 74
3-3 Time taken to solve algorithms for increasing forest size 98
3-4 Number of juries (out of 100) which are predicted to return a guilty verdict according to out-of-sample model, if optimized for guilty verdict 102
3-5 Performance on synthetic data 104

4-1 Example 1 131
4-2 Example: tree ensemble formulation is not ideal 134
4-3 Time taken to solve for single tree with varying size of assignment and tree size 137
4-4 Time taken to solve forest 138
4-5 Minimizing distance travelled 142
4-6 Maximizing number of patients served 144
4-7 Distribution of wait and distance using tree approximation 146
4-8 2 way online matching problem 148
4-9 3 way matching problem 150

5-1 Shipping rates for the TD3 route (between Middle East and Japan) illustrating the variability in shipping rates 155
5-2 An example network for the perfect information problem 173
5-3 How the profit obtained by a broker changes as a function of the number of cargoes considered 179
5-4 Number of cargoes considered perfect information comparison 181
5-5 Comparison in run-time differing number of cargoes 182
5-6 Illustration of the possible transitions for the independent discrete errors around a given forecast 184

A-1 Feature importance as evaluated by boosted trees 199
A-2 Estimating treatment effect for changing sample size 204

B-1 Hazard rate dominance for Y_i as the number of trees sampled (n) grows 216
B-2 In forest vs out of forest behavior 219
B-3 Feature importance random forest 220

C-1 example of how to solve problem with at port 1 both cargoes available 223

List of Tables

2.1 NBA data features 56

2.2 Regression of ticket and market features and the average historical price difference on price 57

2.3 Probit regression of price, ticket and market features and price residual (from IV model) on selling outcomes 57

2.4 Predictive accuracy and miscalibration on NBA data 58

2.5 Comparison of backtesting profit between methods with LGBM IV baseline (limited to purchasing 50 tickets a day) 61

3.1 Examples of optimization problems with an uncertain relationship between actions and outcomes 66

3.2 Variables of housing dataset 90

3.3 Predictive accuracy house sales prices 91

3.4 Variables of our dataset 95

3.5 Predictive jury verdict averaged over 20 data splits 95

3.6 Cross-validation performance on the jury dataset 99

3.7 Cross-validation performance on the housing dataset 99

3.8 Difference in sale price of the property chosen using random forest (RF) vs linear regression as evaluated on OOS XGBoost 103

3.9 Difference in sale price of property chosen using random forest (RF) vs linear regression as evaluated on OOS random forest 103

5.1 Comparison of run-time for differing numbers of cargoes 181

A.1 Predictive accuracy on NBA data iv 200
A.2 Difference between revenue at prescriptive price $\hat{T}$ and true optimal price T^* 205

B.1 Cross-validation performance on jury dataset 218
B.2 Cross validation performance on housing dataset 218
B.3 Standard deviation out of forest price for housing dataset 219

C.1 Autocorrelation for different loading areas over multiple time lags . . 224

Chapter 1

Introduction

Recently, there has been a lot of interest in the business community in utilizing data to make better decisions. One of the drivers behind this interest comes from the increasing availability of data that businesses have at their disposal. Data is being collected and stored at ever increasing rates, with the volume of business data estimated to double every 1.2 years (Meloni (2018)). This has provided many new opportunities for academia and industry, and has lead to a surge in research in data driven algorithms. Recent advances in machine learning by the computer science community has lead to significant progress in *predictive analytics*— using data to make *predictions* about an outcome. Notable high profile successes include detecting cancer from scans and images (Kourou et al. (2015)), using natural language processing to understand user sentiment leading to development of smart assistants (Ram et al. (2018)), and advances in image classification which have enabled self-driving car technology (Bojarski et al. (2016)). Less progress has been made in *prescriptive analytics* — using data to make *decisions*. Some illustrative examples studied in this thesis include using sales data to make pricing decisions for tickets on the resale market (chapter 2), using jury trial data to decide how to select juries in a legal setting (chapter 3), using historical routing data to make decisions about how to route nurse practitioners for a health care on-demand start up (chapter 4), and using shipping rate data to route oil tankers (chapter 5).

Prescriptive analytics, is a relatively new field with many interesting practical

challenges which have not yet been addressed. Some common challenges which are studied in detail in this thesis include model specification and accurately learning the relationship between decisions and outcomes from data, which can sometimes be obscured by confounding factors. Another challenge is making complex decisions in a data-driven environment. In particular, the interplay between decisions which are combinatorial in nature and non-linear machine learning functions leads to some difficult optimization problems. Finally making decisions in an online setting where decisions are made and information is learned over time also introduces several difficulties. Each of these issues is challenging in its own right, but often a real-world problem will incorporate many of these elements. This requires the development of general algorithms which are flexible and adaptable across a range of domains. Solving these problems provides the unique opportunity to combine recent ideas from machine learning, operations research, and econometrics to develop new algorithms which not only have high predictive accuracy but are able prescribe what actions should be taken to provide beneficial outcomes.

A key component of prescriptive analytics is accurately estimating treatment effects. To have any confidence in decisions made using data, it is important to accurately estimate the effect of the decision on the outcome. For example, many revenue management and pricing models use observational data on historical sales to infer demand models, which are subsequently important for pricing decisions. Many of these models used in this setting are able to make accurate predictions over the data trained on, but aren't necessarily able to identify the effect on demand of changing the price. Despite their widespread use in practice, optimizing using these models is risky as it may result in poor pricing decisions, as highlighted by Bertsimas and Kallus (2016). One common issue is hidden confounding factors — whereby there is a variable which is not in the dataset which affects both the pricing decision and the realized demand. This can obscure the treatment effect. This is common even in large observational datasets, as it is often impractical to collect data on absolutely all the variables which may be influential in the decision being made. We provide an instrumental variable approach combined with machine learning techniques in chapter 2,

which addresses these issues. Although we study a pricing example, the applications of this methodology are much broader, with treatment effect estimation in healthcare and in particular personalized medicine being an obvious application of this research.

Another common challenge in prescriptive analytics is model specification. Oftentimes a practitioner has data on previous decisions taken in different situations, and can observe the corresponding outcomes, but doesn't know how precisely how they relate to each other. A traditional approach is to use a parametric model to capture this relationship. These models often make restrictive assumptions on the functional form of the relationship. Similar to the treatment effect estimation issues discussed earlier, the resulting decisions made from these models may be compromised if these assumptions are not met. With increasing availability of high-dimensional large-scale datasets, there is often sufficient data to discover complex non-linear relationships and interactions between variables. Non-parametric machine learning approaches offer a more flexible framework and are able to model a variety of real-world relationships significantly more accurately than parametric methods. Furthermore, there are regularization techniques to prevent overfitting. These methods require minimal feature engineering, which has lead to a recent boom in popularity. In particular, in this thesis we provide a focus on using tree ensembles including random forests (Breiman (2001)), and boosted trees (Freund and Schapire (1995); Chen and Guestrin (2016)). Despite the recent hype around neural networks and deep learning, which have been very successful in domains such as image classification and natural language processing, tree ensemble methods remain the gold standard in many practical business settings. `XGBoost`, a popular boosted tree algorithm, was used in the winning entry for over half of all Kaggle competitons in 2015 (Chen and Guestrin (2016)). In chapter 2, we show how to incorporate non-parametric machine learning functions within a framework for estimating heterogenous treatment effects. In chapter 3 we show how to solve optimization problems where the objective is given by a tree ensemble function. Chapter 4 builds off this foundation as we show how to solve approximate dynamic programming problems where the cost-to-go function is approximated by trees/tree ensembles. In all cases, we demonstrate an ability to improve upon

traditional parametric functions over a range of real-world datasets.

Yet another common challenge encountered in prescriptive analytics is complex decision making settings. Many operations management and operations research problems have multi-dimensional decisions, which contain both continuous and discrete components and are constrained in some way. This can be compared and contrasted with *predictive analytics* settings where often the decision being made is very simple — is this image a dog or a cat? Some examples of complex decisions studied in this thesis include the combinatorial challenges of routing nurse practitioners to patients (chapter 4), oiltankers to cargoes (chapter 5), and selecting juries from a larger pool of potential jurors (chapter 3). In these settings the practitioner isn't able to tractably enumerate all possible decisions but must instead solve an optimization problem. Well studied, tractable algorithms often exist for solving these kinds of problems when the objective is linear. However, when the objective function includes a non-linear machine learning function, relating the decisions made to the outcomes desired, this becomes significantly more difficult and new methods need to be developed. In chapter 3 we provide mixed integer optimization (MIO) formulations and approximate methods for solving these problems. This provides a lot of flexibility in what constraints are incorporated into the problem, and can leverage significant recent improvements in MIO solvers. In chapter 4 we show how to solve these problems in an approximate dynamic programming setting with assignment type constraints. In chapter 5 we show how these challenges can be overcome in the tramp shipping context where the complexity arises from the combinatorial nature of which cargoes are available.

Finally another common feature when making data-driven decisions, is that the problem is dynamic and changes over time. More specifically we look at online optimization problems where sequential decisions need to be made over a time horizon as new information is revealed. Examples studied in this thesis include tramp shipping, where oil tankers need to choose which cargo to choose from a set of dynamically available cargoes every time they come in to port (chapter 5). Similarly chapter 4 concerns routing nurse practitioners to patients and dynamically updating routes as

new patients become available. These problems are difficult because of the curse of dimensionality — the size of the state space is often extremely large. For instance, there is an exponential number of different combinations of nurse practitioners and patients which might be available at any time. Traditional dynamic programming approaches require calculating the value of each state, and as there are so many, this is a formidable task. In chapter 4 we study approximate methods where machine learning techniques are used to learn the value function. In chapter 5 we study a ranking algorithm that can be used to solve the dynamic programs that arise in the maritime shipping setting. Both these algorithms are tractable and lead to algorithms that can be practically implemented as shown in various numerical simulations, relative to traditional DP approaches which are prohibitively slow.

Another common theme in chapters 4 and 5, is applications in dynamic vehicle routing. Effecting routing algorithms have a significant impact on the operational efficiency and competitiveness of many companies in traditional logistics industries and beyond. With the success of ride sharing apps such as Uber and Lyft, there is a growing expectation of immediate service. This has led to many companies, both business-to-business and business-to-consumer, trying to improve their response rates and serve customers in real time. Unlike many traditional transportation models where typically more information is known in advance, these systems are highly dynamic, requiring new modeling techniques to enable companies to perform well in this setting. The applications in chapters 4 and 5 both fit this paradigm. Our industry collaborator in chapter 4 provides a guarantee that all patients will be served within an hour of requesting service, so the horizon over which patients are known is very limited. Similarly, in chapter 5, our collaborator brokers ships on the spot market. The spot market is characterized by short lead times where the time between the cargo being listed and the cargo being picked up is short. Again, this results in brokers having limited foresight about what will happen in the future, making planning and routing decisions difficult.

The ability to overcome the aforementioned challenges, using the methods discussed in this thesis will help unlock many opportunities for businesses to use their

data to make effective decisions.

1.1 Contributions

The main contributions of this thesis concern addressing challenges in the field of prescriptive optimization. These include methodological and applied contributions. The majority of the chapters in this thesis are motivated through work with companies including a large ticket reseller (chapter 2), an on demand healthcare start up (chapter 4), and a maritime shipping company (chapter 5). We show how specific applied business problems these companies face can be solved using analytics. We also abstract these problems to a more general setting, and provide methodological contributions which are relevant to a wider class of problems. We provide rigorous theoretical analysis of the properties of these algorithms. By developing theory which is motivated by current business issues, we discover new and interesting problems, which are relevant to both industry and academia. We next describe in more detail the contributions of each part of the thesis.

In chapter 2, we provide a data-driven study of the secondary ticket market. In particular, we are primarily concerned with accurately estimating price sensitivity for listed tickets. In this setting there are many issues including endogeneity, heterogeneity in price sensitivity for different tickets, binary outcomes and non-linear interactions between ticket features which make the estimation problem challenging. We make several methodological contributions— to the best of our knowledge, we are the first to propose a semi-parametric model for measuring heterogeneous treatment effects using the concept of orthogonalization in the classification setting. To accomplish this goal, we develop a novel loss function that allows us to estimate the treatment effect directly. This loss function can be incorporated into a wide range of off-the-shelf machine learning methods and allows us to leverage the power of cutting edge algorithms. In particular, we show how to integrate this loss function into `lightgbm`, a widely used and highly accurate gradient boosting method. We are able to show that this loss function is consistent, under mild assumptions, on the func-

tional form of the true treatment effect and provide a theoretical rate of convergence in finite sample. We show that in the presence of hidden confounders, instrumental variables can be incorporated. Over a wide range of synthetic data experiments, we show how this approach beats state-of-the-art machine learning and causal inference methods for estimating treatment effects in classification tasks. Moreover, we show how to embed this estimation procedure within a novel market making algorithm. Using NBA ticket listings from the 2014-2015 calendar year, in pricing simulations we show our proposed method is able to achieve an 11% ROI by buying and selling tickets, while existing techniques for estimating price sensitivity are not profitable. This shows that in an industry with slim margins, even slight misspecification of the demand model can result in significant losses. We observe a small but significant improvement from 7% to 11% ROI when using a time-varying pricing policy relative to a constant pricing policy, suggesting there is improvement to be made by considering the perishable inventory effects of tickets. We would also like to note that our approach for heterogeneous treatment estimation methodology in classification settings can be generalized beyond the scope of ticket reselling. For example, in medicine there are many outcomes which are binary in nature (occurrence of a health event such as a heart attack or cancer) and treatments that need to be personalized according to a patient's symptoms and their corresponding health profile. The approach introduced in this chapter could also be used in other revenue management settings where there is a high degree of differentiation and/or personalization of a product, leading to a classification model that is more accurate than an aggregate demand model.

In chapter 3, we show how to solve optimization problems with random forest objective functions and general polyhedral constraints. We show that it is possible to formulate this problem as a MILP and show this formulation can be decomposed and solved iteratively using Pareto-optimal Benders cuts. Although the subproblems contain complicated binary logic, we provide a naturally integer subproblem formulation. This in turn allows us to consider the dual formulation and hence consider Benders cuts. Second, we provide analytical guarantees on an approach that approximates a large scale random forest optimization problem by optimizing over a smaller forest.

We show that the sub-optimality of the approximate solution decays exponentially with the number of trees in the smaller random forest, under certain assumptions on how trees are generated. Third, for very large problem instances, we propose heuristic algorithms which optimize over smaller forests using an approach inspired by cross-validation. We derive analytical upper and lower bounds on the performance of the optimal solution from the heuristic. Finally, we explore the performance of these algorithms using two original case studies with real data; a property investment and a jury selection case study. We show that our approaches outperform benchmarks optimizing other machine learning objective functions.

In chapter 4, we study a new problem arising from a collaboration with a start up, focusing on delivering on-demand health care services. In this setting, nurse practitioners need to be dynamically routed to patients' houses as service requests are received. We show how to model the nurse practitioner routing problem using Markov Decision Process and solve using Approximate Dynamic Programming. Furthermore, we show how to formulate general ADP's with combinatorial action spaces and non-linear cost-to-go functions using a tree or tree ensemble approximation. We present a novel, Mixed Integer Optimization formulation for optimizing over a tree. We prove this formulation is ideal. We extend to include assignment constraints and binary inputs and explore properties of these formulations. Using extensive numerical experiments, we show these formulations are able to outperform other formulations in terms of speed to find the optimal solution. We also show that the tree approximation performs well relative to other ADP approximations for the nurse practitioner problem using numerical experiments.

In chapter 5, we propose a Markov Decision Process (MDP) model for the tramp shipping problem that captures the dynamic and stochastic nature of spot cargo availability. This captures how the brokers consider potentially large sets of cargoes in each period. To the best of our knowledge, we are the first to model a tramp shipping firm that operates only in the spot market. Unfortunately, in general, our proposed formulation is hard to solve using a traditional dynamic programming approach. This is due to the state space of the problem being exponential in the number of cargoes

considered. We propose a novel methodology for solving this MDP in a tractable way, by introducing a ranking algorithm which is polynomial in the number of possible cargoes at each port. We prove this algorithm is equivalent to solving the proposed DP. Through real-world data, we build a simulation environment which allows us to evaluate the performance of this algorithm relative to several other MDP approximation methods. We show that our ranking algorithm outperforms several benchmarks and the average performance of ships operating on the spot market in practice by between 4% and 32% depending on the number of cargoes the broker is able to consider. For the case of independent shipping rates, and when these shipping rates are discrete, we propose an extension of the ranking algorithm. We show that this algorithm is polynomial in terms of the number of cargoes and discrete rate levels. We show that we can extend the ranking algorithm to be applied in the infinite horizon problem using linear programming or alternatively a modified policy iteration algorithm. Not only is this interesting from an analytical perspective but it is also a practical objective for ship brokers to optimize, since the solution to the infinite horizon problem presents a steady state view of the system which is useful to the shipping company for medium term planning.

Chapter 2

Pricing for heterogeneous products: analytics for ticket reselling

2.1 Introduction

Ticket marketplaces are a rapidly growing industry with listings for over 10 million events per month (Erskine (2015)). They host an exchange for sellers who wish to sell tickets to interested buyers. Most of these tickets have been previously purchased and are not sold from the primary seller, but rather they are being resold. Although these marketplaces have an abundance of observational data on how sellers price tickets, understanding market dynamics remains a challenge. Many firms have difficulty in estimating demand and its sensitivity to price. Additionally, researchers have highlighted the high elasticity of demand in secondary markets, and in particular how it varies heterogeneously across different tickets (Diehl et al. (2015), Jiaqi Xu et al. (2019)). Through our collaboration with our industry collaborator, we notice that estimating price sensitivity in demand is a particularly challenging problem for the following reasons:

-*Heterogeneity in treatment effect* - Different seats have different sensitivity to price. In the data, we observe that tickets that are likely to sell tend to be less price sensitive. However, even for tickets which have similar estimated probabilities of selling, there are still differences in price sensitivity, suggesting that there is further

interaction between the ticket features and the price which our model aims to capture. Heterogeneous treatments are more difficult to estimate than homogeneous treatments due to the relative sparsity of individuals who react in the same way.

-*Hidden confounding factors* - A common issue firms face when trying to make inferences on data is confounding by hidden variables, which influence both the treatment (price) and the outcome (demand). This often occurs because firms are not able to exhaustively record, or may not have access to, all variables which are relevant to the pricing decision being made. This is true for our industry collaborator, who has a very rich dataset with thousands of explanatory variables but econometric tests still suggest the presence of endogeneity (See section §2.6.2). These hidden confounding factors can lead to biased estimation of the parameters of interest.

-*Large, high dimensional datasets* - The dataset from our industry collaborator contains millions of tickets, which potentially allows for the estimation of very rich models with complex interaction effects. Recent advances in machine learning and non-parametric estimation have enabled these effects to be captured, leading to models with high predictive accuracy. Despite the success in improving predictive accuracy, less progress has been made on using machine learning for inference or treatment effect estimation. The high dimensionality of the data also justifies the use of machine learning models that are able to use regularization to avoid overfitting.

- *Continuous treatments and Binary outcomes* - The majority of the literature on causal treatment effect estimation focuses on binary or discrete treatments and continuous outcomes (Imbens and Rubin (2015)). Binary outcomes make the problem more challenging because of the non-linear response function required to transform the response variables into a binary outcome. Similar to heterogeneous treatments, continuous treatments are difficult because of the sparsity of individuals who receive the same or similar treatments relative to the binary case. However, in our setting price is a continuous variable. Furthermore, due to the heterogeneous nature of tickets and the market, demand must be estimated on a per ticket basis using the binary outcome of whether a ticket sold. Unlike a primary ticket seller, and in contrast to many traditional revenue management settings, at any given time there are only a few

tickets on the market relative to the total number of tickets in the stadium where the game will take place. For example, with our industry collaborator there is on average there is between 5% and 10% of the stadium available for sale at a given time. This further amplifies the effects of heterogeneity and makes it hard to pool similar tickets together into an aggregate demand model.

In this work, our primary goal is to accurately isolate and capture the treatment effect of price within a prediction framework that estimates the probability of an individual ticket selling. Our secondary goal is to integrate this model into a prescriptive trading strategy for buying and selling tickets in an active marketplace. Accurately estimating the causal price effects is very important when selling tickets, as incorrect estimates of the price sensitivity may lead to sub-optimal pricing decisions (Bertsimas and Kallus (2016a)). The knowledge of how to price tickets on its platform offers a range of potential opportunities for our collaborator, both in terms of understanding sellers on their platform and in developing new products to offer them. One potential application would be to offer sellers a bid at the time of listing wherein our industry collaborator would advance the seller a guaranteed payment and then take on the risk of selling the inventory. This offers security to the seller as they no longer bear the risk that their ticket will not sell and the burden of managing the pricing.

Contributions: In this chapter, we make several methodological contributions. To the best of our knowledge, we are the first to propose a semi-parametric model for measuring heterogeneous treatment effects using the concept of orthogonalization in the classification setting. To accomplish this goal, we develop a novel loss function that allows us to estimate the treatment effect directly. This loss function can be incorporated into a wide range of off-the-shelf machine learning methods and allows us to leverage the power of cutting edge algorithms. In particular, we show how to integrate this loss function into `lightgbm`, a widely used and highly accurate gradient boosting method. We are able to show that this loss function is consistent, under mild assumptions, on the functional form of the true treatment effect and provide a theoretical rate of convergence in finite sample. We show that in the presence of hidden confounders, instrumental variables can be incorporated. Over a

wide range of synthetic data experiments, we show how this approach beats state-of-the-art machine learning and causal inference methods for estimating treatment effects in classification tasks. Moreover, we show how to embed this estimation procedure within a novel market making algorithm. Using NBA ticket listings from the 2014-2015 calendar year, in pricing simulations we show our proposed method is able to achieve an 11% ROI by buying and selling tickets, while existing techniques for estimating price sensitivity are not profitable. This shows that in an industry with slim margins, even slight misspecification of the demand model can result in significant losses. We observe a small but significant improvement from 7% to 11% ROI when using a time-varying pricing policy relative to a constant pricing policy, suggesting there is improvement to be made by considering the perishable inventory effects of tickets. We would also like to note that our approach for heterogeneous treatment estimation methodology in classification settings can be generalized beyond the scope of ticket reselling. For example, in medicine there are many outcomes which are binary in nature (occurrence of a health event such as a heart attack or cancer) and treatments that need to be personalized according to a patient's symptoms and their corresponding health profile. The approach introduced in this chapter could also be used in other revenue management settings where there is a high degree of differentiation and/or personalization of a product, leading to a classification model that is more accurate than an aggregate demand model.

Managerial insights: Our results present valuable insights on the state of the marketplace and show there is potential to improve overall marketplace efficiency for our industry collaborator. We estimate that 12% of tickets are more than 10% under priced relative to optimal pricing. This is a significant arbitrage opportunity resulting in surplus that is currently captured by 3rd party market makers, a practice our industry collaborator could reduce without hindering liquidity. Similarly, we estimate that 14% of tickets are currently more than 10% overpriced relative to optimal pricing. This suggests there is a significant opportunity to help sellers through "better" pricing practices. Using our suggested algorithms, it is possible to correctly estimate whether a ticket will sell at a given price with 91% accuracy, an insight which is valuable in

helping our industry collaborator develop products which enhance the user experience. We show that IV probit methods previously used for price estimation of tickets in the resale market are significantly less accurate and potentially misspecified relative to our proposed approach. More broadly, we hope this study highlights the potential difficulties involved in causal price sensitivity estimation, and implications on pricing practice.

2.2 Literature review

Our work lies at the intersection of three streams of literature; pricing for events, data-driven pricing and causal inference.

2.2.1 Causal inference

In the operations management community, there is growing awareness of the need to identify the causal relationships of treatments or interventions that are used in making business decisions. In the presence of endogeneity, an instrumental variable is required to identify the treatment effect. Starting with continuous outcomes (Wright (1928); Reiersøl (1945)), there is a body of work which extended instrumental variable models to binary outcomes using linear probit models (Rivers and Vuong (1988); Lee (1981); Amemiya (1978)). These models are often referred to in the literature as control function approaches. Similar to our setting, these models have a continuous treatment but treatment effects are homogeneous. Due to recent advances in machine learning and advantages in utilizing high dimensional data, there have been extensions to non-parametric models with instrumental variables and binary outcomes (Blundell and Powell (2003, 2004)). Our main contribution over this literature is showing the effectiveness of an orthogonalization step in the estimation, which is beneficial regardless of whether confounding is present, and the heterogeneity in effect estimation. We outline the estimation procedure of Rivers and Vuong (1988) and Blundell and Powell (2003) in more detail in section §2.4, where we use these approaches as benchmarks in simulation experiments. For a comprehensive review of instrumental variable models

with binary outcomes, we refer the reader to Imbens and Wooldridge (2007).

Another stream of literature for estimating treatment effects, with observational data uses Inverse Propensity Weighting (Horvitz and Thompson (1952)). The imbalance in the observational data is corrected by dividing through by propensity score, the conditional probability of being assigned a treatment given the available features. Estimating the probabilities can be practically difficult to accomplish, and these models are known to be sensitive to misspecification in the propensity score. In addition, these papers typically assume no hidden confounding variables, which the instrumental variable approach is able to overcome. Once the imbalance has been corrected, the average treatment effects can be calculated, while methods such as (Xie et al. (2012); Raudenbush and Bryk (1986)) use stratification of the data to provide heterogeneous treatment effects for particular groups. Our approach is able to identify more granular treatment effects on the ticket level. Matching approaches can also be used for binary outcomes (Austin and Stuart (2017)). For further reading on causal inference and estimation of treatment effects, we refer the reviewer to Imbens and Rubin (2015).

A relatively recent stream of literature for estimating treatment effects is orthogonalized or double machine learning methods. These methods aim to isolate the effects of the treatment on the outcome by removing the conditional effects of the control variables. The methodology was first introduced by Robinson (1988) for estimating parametric components in partially linear models. Chernozhukov et al. (2018) provide examples of estimating (homogeneous) treatment effects in the presence of high dimensional controls. Oprescu et al. (2018) provide some extensions of this approach to a heterogeneous treatment effect estimation using random forests. Nie and Wager (2017) develop a specific loss function for the regression setting of heterogeneous treatment effect estimation with binary treatments, using the orthogonalized outcome and treatment. They show improvements using boosted trees and lasso regression compared to single stage analogues. In contrast, our approach provides novel loss functions and algorithms for the classification setting for continuous rather than binary treatments.

2.2.2 Ticket pricing for sporting events

There have been several papers studying pricing for event sales for sports events (Bouchet et al. (2016), Barlow (2000), Duran et al. (2012),Sainam et al. (2010), Kemper and Breuer (2016), Jiaqi Xu et al. (2019)). These papers focus on the problem faced by primary sellers rather than resellers. They typically assume demand can be aggregated by section in the venue of the event. While this is a reasonable assumption to make in the primary sellers case due to the large number of tickets they are selling, this is not as reasonable in the reselling case where the tickets being sold at any time are extremely sparse.

With the growing popularity of secondary ticket markets, there has been recent academic interest in this area. Similar to our work, Sweeting (2012) and Zhu (2014) develop dynamic pricing models for sellers of a single ticket in the MLB resale market, but are more concerned with capturing the market behavior than prescribing the optimal prices. They develop probit and multinomial logit models respectively to model the effect of price on the probability of selling an individual ticket. Zhu (2014) offers extensions to the case of strategic consumers who potentially defer purchasing decisions. Diehl et al. (2015) studies the NFL resale market using an aggregate demand model and IV regression. These models use traditional econometric techniques and transformations of linear specifications, which aren't necessarily able to capture the complex interactions which exist in large scale data, relative to the machine learning based techniques we develop.

2.2.3 Data-driven pricing

There are recent papers on data-driven pricing problems which incorporate high dimensional features of the product or customer. For example, Cohen et al. (2016), Javanmard and Nazerzadeh (2016),Ban and Keskin (2017), Ban and Keskin (2017) study online dynamic pricing settings where the demand has to be learned over time, and the goal is to minimize regret. Ferreira et al. (2015) predict demand using a random forest and optimize price to maximize revenue in online retail. Further work

on optimizing tree ensembles has been done by Biggs and Hariss (2018). These methods do not explicitly consider confounding in the pricing decision. The issues with ignoring confounding in pricing is well documented in Bertsimas and Kallus (2016a). For a comprehensive review of pricing in the operations management literature, we refer the reader to Talluri and Van Ryzin (2006).

From the economics literature, combining pricing with heterogeneous treatment effect estimation under confounding can be found in Chernozhukov et al. (2017). This paper builds on Chernozhukov et al. (2018) for demand estimation with high dimensional controls. The paper considers a partial linear model with explicit interaction terms between price and features of the product. The paper assumes a continuous aggregate demand model rather than a discrete classification model.

2.3 Heterogeneous treatment effect estimation for classification

In this section, We consider a latent variable model that is a semi-parametric variant of the model proposed in Rivers and Vuong (1988), which explicitly models the confounding effect of ticket features on price. We assume access to n identically and independent samples $W_i = (Y_i, X_i, T_i, Z_i), i = 1, ..., n$, where $Y_i \in \{0, 1\}$ is a binary outcome, $T_i \in \mathbb{R}$ is the treatment, $X_i \in \mathcal{X} \subseteq \mathbb{R}^d$ is the control, and $Z_i \in \mathbb{R}^k$ is a set instrumental variables. We propose the following model where Y^* is an unobserved latent random variable such that:

$$Y^* = g(X) + \tau(X)T + \epsilon \tag{2.1}$$

$$T = m(X) + \beta^T Z + \upsilon \tag{2.2}$$

$$Z = h(X) + u \tag{2.3}$$

$$Y = \begin{cases} 1, & \text{if } Y^* > 0 \\ 0, & \text{if } Y^* \leq 0 \end{cases} \tag{2.4}$$

$$\begin{pmatrix} \epsilon \\ \upsilon \end{pmatrix} \sim N\left(\begin{matrix} 0 \\ 0 \end{matrix}, \begin{pmatrix} 1 & \rho\sigma_\upsilon \\ \rho\sigma_\upsilon & \sigma_\upsilon^2 \end{pmatrix} \right), \; u \sim N(0,1), \; \mathbb{E}[u\upsilon|X], \; \mathbb{E}[u\epsilon|X] = 0 \tag{2.5}$$

In the ticket selling context, Y is the observation of whether the ticket is sold or not. X corresponds to a set of features with respect to the ticket and the market state (for a more detailed description, see section §(2.6.1). T is the price chosen for that ticket. The heterogeneous treatment effect is captured through $\tau(X)$, which can be interpreted as the price sensitivity dependent on the features of the ticket. The term $m(X)$ can be interpreted as the fair market value of the ticket (the portion of the ticket's price that can be explained by the features of the ticket), while v can be interpreted as the idiosyncratic choice by the seller of how to price the ticket relative to the fair market value. The term $g(X)$ corresponds to the contribution to the likelihood of selling of the ticket features.

Finally, the model can suffer from omitted variable bias, where there are unobserved variables which affects both Y and T. These endogeneous factors cause correlation between the residual errors ϵ and υ, which can cause bias in the estimate of the treatment effect $\tau(X)$. To overcome this, suitable instruments Z are required, which only affect Y through the interaction with T. Through variation of Z, we can observe the impact of changing the treatment T on Y. These relationships are captured through the condition $\mathbb{E}[u\upsilon|X]$ and $\mathbb{E}[u\epsilon|X] = 0$. The term $h(X)$ captures possible correlation between the instruments and the features of the ticket. We go over the specific set of instruments we use in the ticket selling case study in section §2.6.2.

The model in (2.1- 2.5) is difficult to estimate due to the semi-parametric form of equations (2.1 - 2.3). The non-linear terms $g(X), \tau(X), m(X), h(X)$ are challenging to estimate jointly using maximum likelihood estimation. Simpler models where these terms are linear and the treatment is homogeneous, such as in Rivers and Vuong (1988), can perform poorly when the true models are non-linear as is often the case with real data. Fully non-parametric models, such as boosted trees and neural networks, are often able to achieve good predictive accuracy, but do not explicitly

exploit the structure of the model to estimate $\tau(X)$, leading to poor approximate estimates of the heterogeneous treatment effect.

Our model provides a general approach that works for heterogeneous treatment effect estimation in the presence of endogeneity using instrumental variables. However, we note that this method can also be applied in the special case where ϵ and υ are independent (no endogeneity), in which case an instrumental variable Z is not required. We show in section §2.4 strong empirical evidence that this methodology is more effective in the uncorrelated case than other machine learning approaches which do not use orthogonalization.

2.3.1 Method

We follow the literature on double/orthogonalized machine learning to isolate the causal effects of the treatment on the outcome by removing the conditional effects of the control variables. Machine learning algorithms are known to be good predictors, but this literature also shows they can be adapted to be good treatment effect estimators that are robust to confounding. Using orthogonalization, we derive a novel loss function for treatment effects for classification. We first partial out the effect of X from T to obtain the orthogonalized regressor $T - \mathbb{E}[T|X]$. Furthermore, we partial out the effect of X on Y^* to obtain an orthogonalized outcome $Y^* - \mathbb{E}[Y^*|X]$. Since $\mathbb{E}[Y^*|X] = g(X) + \tau(X)m(X)$ and $\mathbb{E}[\epsilon|X] = 0$, we can rearrange (2.1) and (2.2) as follows:

$$Y^* - \mathbb{E}[Y^*|X] = \tau(X)(T - \mathbb{E}[T|X]) + \epsilon. \tag{2.6}$$

This new expression has the advantage of eliminating $g(X)$ and thus removing the direct effect of confounding. In the classification setting, since Y^* is a latent variable, $\mathbb{E}[Y^*|X]$ is generally not known, but $\mathbb{E}[Y|X]$ can often be approximated using non-parametric machine learning techniques. Under some assumptions that we will present below, we next link $\mathbb{E}[Y^*|X]$ to $\mathbb{E}[Y|X]$. In what follows, we take $k = 1$

(one instrumental variable) but all the results are generalizable in the case of $k > 1$.

Proposition 1. *If* $\begin{pmatrix} \epsilon \\ \upsilon \end{pmatrix} \sim N\left(\begin{matrix} 0 \\ 0 \end{matrix}, \begin{pmatrix} \sigma_\epsilon^2 & \rho\sigma_\epsilon\sigma_\upsilon \\ \rho\sigma_\epsilon\sigma_\upsilon & \sigma_\upsilon^2 \end{pmatrix}\right)$ *, and* $u \sim \mathcal{N}(0, \sigma_u^2)$ *is uncorrelated with* ϵ, υ*, then:*

$$\mathbb{E}[Y^*|X] = \sqrt{\sigma_u^2\tau(X)^2\beta_z^2 + \sigma_\epsilon^2 + 2\rho\tau(X)\sigma_\epsilon\sigma_\upsilon + \tau(X)^2\sigma_\upsilon^2} \cdot \Phi^{-1}(\mathbb{E}[Y|X]).$$

The proof can be found in Appendix A.2. We can also decompose $\epsilon = \frac{\rho\sigma_\epsilon}{\sigma_\upsilon}\upsilon + \tilde{\epsilon}$ into orthogonal components, where $E[\upsilon\tilde{\epsilon}] = 0$, and $\tilde{\epsilon}|X \sim N(0, (1-\rho^2)\sigma_\epsilon^2)$ (see Bertsekas and Tsitsiklis (2002)). With this modification, the treatment T is orthogonal to the error $\tilde{\epsilon}$. The expression in Proposition 1 can be to substituted into (2.6) to obtain a new expression for the latent variable:

$$\begin{aligned} Y^* = \Phi^{-1}(\mathbb{E}[Y|X])\sqrt{\sigma_\epsilon^2 + 2\rho\sigma_\epsilon\sigma_\upsilon\tau(X) + (\sigma_\upsilon{}^2 + \sigma_u^2\beta^2)\tau(X)^2} \\ + \tau(X)(T - \mathbb{E}[T|X]) + \frac{\rho\sigma_\epsilon}{\sigma_\upsilon}\upsilon + \tilde{\epsilon} \end{aligned} \tag{2.7}$$

In the remainder of the chapter, we assume for simplicity and without loss of generality that $\sigma_\epsilon = 1$. This is a standard assumption in the IV probit literature (Rivers and Vuong (1988))[1]. For brevity, denote w=$(\beta, \sigma_u, \sigma_\upsilon)$. From expression (2.7), we can derive an appropriate loss function resulting from the normally distributed random component $\tilde{\epsilon}$:

$$\begin{aligned} l(X, T, Y, \tau, \rho, w, \upsilon) = Y\log(\Phi(f(X, T, \tau, \rho, w, \upsilon)) + \\ (1 - Y)\log(1 - \Phi(f(X, T, \tau, \rho, w, \upsilon))) \end{aligned} \tag{2.8}$$

$$with \quad f(X, T, \tau, \rho, w, \upsilon) = \frac{\Phi^{-1}(\mathbb{E}[Y|X])k(X, \tau, \rho, w) + \tau(X)(T - \mathbb{E}[T|X]) + \frac{\rho}{\sigma_\upsilon}\upsilon}{\sqrt{1-\rho^2}} \tag{2.9}$$

$$and \quad k(X, \tau, \rho, w) = \sqrt{1 + 2\rho\sigma_\upsilon\tau(X) + (\sigma_\upsilon{}^2 + \sigma_u^2\beta^2)\tau(X)^2} \tag{2.10}$$

[1] Although in general σ_ϵ is not identifiable, it is not necessary to estimate the treatment effect. In the case it is not equal to one, we estimate a scaled treatment effect $\hat{\tau}(X) = \frac{\tau(X)}{\sigma_\epsilon}$, which reflects the change in probability with treatment

We note that most of these nuisance parameters can be estimated prior to optimization of this loss function using machine learning algorithms. $\hat{r}(X) \approx \mathbb{E}[Y|X]$ can be estimated using non-parametric classification prediction methods of Y on X. An estimator $\hat{q}(X) \approx \mathbb{E}[T|X]$ can be found by non-parametric regression of T on X , $\hat{h}(X) \approx \mathbb{E}[Z|X]$ by Z on X while $\hat{u}$ can be calculated as the residuals.

In the case where $m(X)$ is linear, linear regression of T on X and Z can be used for estimates $\hat{m}(X)$ and $\hat{\beta}$, while $\hat{v}$ are the residuals. In the general non-linear case, an additional orthogonalization step can be used on (2.2):

$$T - \mathbb{E}[T|X] = \beta(Z - h(X)) + \upsilon \tag{2.11}$$

So $\hat{\beta}$ can be calculated by regressing $T - \hat{q}(X)$ on $Z - \hat{h}(X)$, while again $\hat{v}$ are the residuals. In either case, σ_v and σ_u can be estimated as $\hat{\sigma_\upsilon} = \frac{1}{n-1}\sqrt{\sum_i \hat{v}_i^2}$, $\hat{\sigma_u} = \frac{1}{n-1}\sqrt{\sum_i \hat{u}_i^2}$. The above approach is summarized in the following algorithm:

Algorithm 1 Two stage estimation for classification algorithm with instrumental variables

1: *Fit* $\hat{r}(x) \approx E[Y|X]$, $\hat{q}(X) \approx \mathbb{E}[T|X]$, $\hat{h}(X) \approx \mathbb{E}[Z|X]$ *via appropriate non-parametric methods, calculate* $\hat{w} = (\hat{\beta},\ \hat{\sigma_u},\ \hat{\sigma_v})$ *and* $\hat{v}$.

2: *Define the approximate log-likelihood function:*

$$\begin{aligned}\hat{l}(X, T, Y, Z, \tau, \rho) = Y \log(\Phi(f(X, T, \tau, \rho, \hat{w}, \hat{v})) \\ + (1 - Y) \log(1 - \Phi(f(X, T, \tau, \rho, \hat{w}, \hat{v})))\end{aligned} \tag{2.12}$$

$$s.t.\quad f(X, T, \tau, \rho, \hat{w}, \hat{v}) = \frac{\Phi^{-1}(\hat{r}(X))k(X, \tau, \rho, \hat{w}) + \tau(X)(T - \hat{q}(X)) + \frac{\rho}{\hat{\sigma}_v}\hat{v}}{\sqrt{1 - \rho^2}} \tag{2.13}$$

and estimate treatment effects via the approximate maximum likelihood estimator:

$$(\hat{\tau}(\cdot), \hat{\rho}) = \arg\min_{\tau,\rho} \left\{ -\frac{1}{n}\sum_{i=1}^{n} \hat{l}(X_i, T_i, Y_i, Z_i, \tau, \rho) \right\} \tag{2.14}$$

In this procedure, the first step learns an approximation for relevant nuisance parameters , while the second step optimizes the approximated empirical log-likelihood loss function. Optimizing this loss function requires finding the optimal function $\hat{\tau}(\cdot)$ and the correlation parameter $\hat{\rho}$. As ρ is one dimensional, we recommend using a line search algorithm to find $\hat{\rho}$. In particular, for a sequence of ρ_k, we find the optimal $\tau(X)_k$ for each, where $\hat{\tau}_k(\cdot) = \arg\min_\tau -\frac{1}{n}\sum_{i=1}^{n} \hat{l}(X_i, T_i, Y_i, \tau, \rho_k, v_i)$ and ρ_{k+1} is chosen by Brent's method (for example), until convergence is achieved.

Various machine learning optimization methods and base functions can be used for estimating the function $\tau(X)$. This step can efficiently be solved by adapting off the shelf methods, using a customized loss function. In the ticket reselling setting, we observe that restricting $\hat{\tau}(\cdot)$ to be a tree ensemble and optimizing the second stage using the `lightgbm` package (Ke et al. (2017)) works well in practice as shown in section §2.6.3. The `lightgbm` package is an example of gradient boosted trees (Friedman (2001)). The wide effectiveness of ensembles are displayed in Kaggle competitions with 60% of winning solutions using gradient boosting implementations (Rogozhnikov and Likhomanenko (2017)). Alternatively, a deep learning approach could be taken by restricting $\hat{\tau}(\cdot)$ to be a neural network and optimizing the second stage loss function using `tensorflow` (Abadi et al. (2016)). For simpler settings, such as when the treatment effect is constant ($\hat{\tau}(\cdot) = \tau$), a one dimensional search of the loss function can be used to find the optimal value (see section §2.4.1 for experiments). For the case where the treatment is a linear function ($\hat{\tau}(X) = \tau' X$), stochastic gradient descent approaches can be used.

Notice that this method presents a general way of estimating heterogeneous treatments for classification tasks, as it can be generalized to other variants of the problem depending on the distributions of ϵ and v. It is possible to derive analogous equations for the case where ϵ and v are not Gaussian using an appropriate CDF $\tilde{\Phi}(\cdot)$. Nevertheless, this CDF might be difficult to calculate in practice. Next, we show consistency of the loss function (2.14).

2.3.2 Consistency and stability of estimator $\hat{\tau}(\cdot)$

The two-stage method we introduce above leads us to determine an approximate maximum likelihood function (2.14) that we optimize. The goal is to ensure it is close to the true maximum likelihood.

In this section, we show that under mild regularity conditions, the optimization of the approximate maximum likelihood function would approximate well, with enough data, the true treatment $\tau(\cdot)$. This result will provide a good justification for the two stage method we introduced. This proof uses standard assumptions in the econometrics literature (Engle (1994)) about consistency, compactness and identifiability of the treatment effect. We note that we do not need to assume identifiability of our loss function, but show in the proof how it follows from our model. Another feature we show, is that we still converge as long $\hat{r}(X)$ and $\hat{q}(X)$ converge (along with the nuisance parameters). Our proof holds for a broad class of treatment effects, in which the treatment can be characterized by finitely many parameters $\tau(X) = t(\theta_1, \cdots, \theta_p, X)$. The theorem below describes this result more rigorously, and also characterizes the rate of convergence.

Theorem 1. *Assume that for the latent variable model in (2.1,2.2,2.3,2.4,2.5), we have $\tau(X)$ as a finitely parametrized function $\tau(X) = t(\theta_1, \cdots, \theta_p, X) = t(\boldsymbol{\theta}, X)$ where $p \in \mathbb{Z}^+$, and t are known. Then we can parametrize the log-likelihood as $l(X, T, Y, Z, \tau, \rho) = l(X, T, Y, Z, \boldsymbol{\Theta})$ where $\boldsymbol{\Theta} = (\boldsymbol{\theta}, \rho)$ and similarly for $\hat{l}, f, k$. Therefore, for iid samples $(X_i, Y_i, T_i, Z_i)_{i=1\cdots n} \in \mathcal{D}_X \times \mathcal{D}_Y \times \mathcal{D}_T \times \mathcal{D}_Z$, define the estimator $\hat{\tau}$ as:*

$$\hat{\tau}(\cdot) = t(\hat{\theta}_1, \cdots, \hat{\theta}_p, X) = t(\hat{\boldsymbol{\theta}}, X) \tag{2.15}$$

$$s.t.\ (\hat{\boldsymbol{\theta}}, \hat{\rho}) = \hat{\boldsymbol{\Theta}} = \underset{\boldsymbol{\Theta} \in \mathcal{D}_\Theta}{\arg\max} \left\{ \frac{1}{n} \sum_{i=1}^{n} \hat{l}(X_i, T_i, Y_i, Z_i, \boldsymbol{\Theta}) \right\} \tag{2.16}$$

We further assume the following:

1. *Compactness: $\boldsymbol{\Theta} \in \mathcal{D}_\Theta$ where $\mathcal{D}_\Theta$ is compact and $\mathcal{D}_X$ is compact.*

2. *Continuity:* $r(X), q(X)$ *are continuous,* $0 < r(X) < 1$ *and* $t(\boldsymbol{\theta}, X)$ *is continuous in* $\boldsymbol{\theta}$ *for every* X*, and is continuous in* X *for every* $\boldsymbol{\theta}$ *with probability 1.*

3. *Consistency of 1st stage:* $\sup_{X \in \mathcal{D}_x} |\hat{r}(X) - r(X)| \xrightarrow{p} 0$, $\sup_{X \in \mathcal{D}_x} |\hat{q}(X) - q(X)| \xrightarrow{p} 0$*,and similarly for all the nuisance parameters.*

4. *Identifiability of treatment effect:* $\boldsymbol{\theta} = \boldsymbol{\theta}_0 \quad \Leftrightarrow \quad t(\boldsymbol{\theta}, \cdot) = t(\boldsymbol{\theta}_0, \cdot)$

Then the estimator $\hat{\tau} = t(\hat{\boldsymbol{\theta}}, X)$ *is a consistent estimator of* $\tau = t(\boldsymbol{\theta}, X)$*, and* $\hat{\rho}$ *is a consistent esimator of* ρ*, as in:*

$$\sup_{X \in \mathcal{D}_X} \|t(\hat{\boldsymbol{\theta}}, X) - t(\boldsymbol{\theta}, X)\| \xrightarrow{p} 0 \qquad \|\hat{\rho} - \rho\| \xrightarrow{p} 0$$

Furthermore, if we have $\sup_{X \in \mathcal{D}_x} n^{1/(2+\delta)} |\hat{r}(X) - r(X)| \xrightarrow{p} 0$, $\sup_{X \in \mathcal{D}_x} n^{1/(2+\delta)} |\hat{q}(X) - q(X)| \xrightarrow{p} 0$*, and similarly for all the nuisance parameters for some* $\delta > 0$*, then, we have:*

$$\sup_{X \in \mathcal{D}_X} \sqrt{n} \|t(\hat{\boldsymbol{\theta}}, X) - t(\boldsymbol{\theta}, X)\| \xrightarrow{p} 0 \qquad \sqrt{n} \|\hat{\rho} - \rho\| \xrightarrow{p} 0$$

The proof can be found in Appendix A.2.

Corollary 1. *Assume that for the latent variable model in (2.1,2.2,2.3,2.4,2.5), we have* $\tau(X)$ *as a finitely parametrized function* $\tau(X) = t(\theta_1, \cdots, \theta_p, X) = t(\boldsymbol{\theta}, X)$ *where* $p \in \mathbb{Z}^+$*, and* t *are known.Then the log likelihood* $\mathbb{E}_{X,T,Y,Z}[l(X, T, Y, Z, \boldsymbol{\theta}, \rho)]$ *has a unique global optimal solution.*

This corollary follows from Theorem 1 and the properties of the log likelihood function (see Lemma 2.2 in Engle (1994)). Through extensive numerical simulations on synthetic data, we show that gradient boosted trees are able to find the unique optimal solution fast.

2.4 Testing the method: experiments on synthetic data

We begin with estimating a constant treatment effect, then we progress to the case of heterogeneous treatment effect, using gradient boosted trees.

2.4.1 Homogeneous treatment effect simulations

We study a case of model (2.1,2.2,2.3,2.4, 2.5) where the functional form of the nuisance parameters is linear:

$$Y^* = \beta_g' X + \tau T + \epsilon, \;\; T = \beta_m' X + \beta_z Z + \upsilon, \;\; Z = \beta_h + u, \begin{pmatrix} \epsilon \\ \upsilon \end{pmatrix} \sim N \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & \rho\sigma_\upsilon \\ \rho\sigma_\upsilon & \sigma_\upsilon^2 \end{pmatrix} \right)$$

The endogeneity is captured by the correlation between ϵ and υ, where $\tau = 2, \rho = 0.5, \;\sigma_\upsilon = 1, \;u, \beta_h \sim N(0,1), \; X \sim N(0, I_d)$ and $\beta_g, \beta_m \sim N(0, I_d)$ but only have 5 non-zero components.

We test against the seminal work by Rivers and Vuong (1988) from the literature on probit models with instrumental variables. It provides an estimation procedure based on the decomposition of $\epsilon = \frac{\rho}{\sigma_\upsilon}\upsilon + \tilde{\epsilon}$ into independent components, where $E[\upsilon\tilde{\epsilon}] = 0$ and:

$$Y^* = \beta_g' X + \tau T + \frac{\rho}{\sigma_\upsilon}\upsilon + \tilde{\epsilon}$$

Rivers and Vuong (1988) proposed the following two stage procedure (`probit_iv`):

1. Regress T on X and Z to find $\hat{\upsilon} = T - (X'\hat{\beta_m} + \hat{\beta}_z Z)$, and $\sigma_\upsilon = \frac{||\hat{\upsilon}||}{\sqrt{n}}$.

2. Probit regression of Y on $X, Z, \hat{\upsilon}$ to obtain estimates of $\hat{\beta}_g, \hat{\tau}, \hat{\lambda} = \frac{\hat{\rho}}{\sigma_\upsilon}$.

This method is different from our proposed approach as it does not have the orthogonalization step to reduce sensitivity to the nuisance parameters.

Figure 2-1a shows how the error in the estimated treatment coefficient changes with the dimension of the non-signal component of the data. We generated 100

datasets with $n = 1000$ data-points each and used `probit_iv` and the proposed two stage method (`two_stage`) to estimate the treatment coefficient for each dataset. To optimize the second stage loss function (2.14), we only require a simple line search to find the correct treatment coefficient. For the first stage, $\hat{r}(x) \approx E[Y|X]$ is estimated using logistic regression while $\hat{q}(x) \approx E[T|X]$ and $\hat{h}(x) \approx E[Z|X]$ are estimated using OLS regression. Figure 2-1a shows the `probit_iv` estimator is unable to accurately estimate the treatment effect when the dimension is high. In this setting, it is beneficial to do the orthogonalization step, which is why `two_stage` is able to more accurately estimate the treatment effect in a larger dimension.

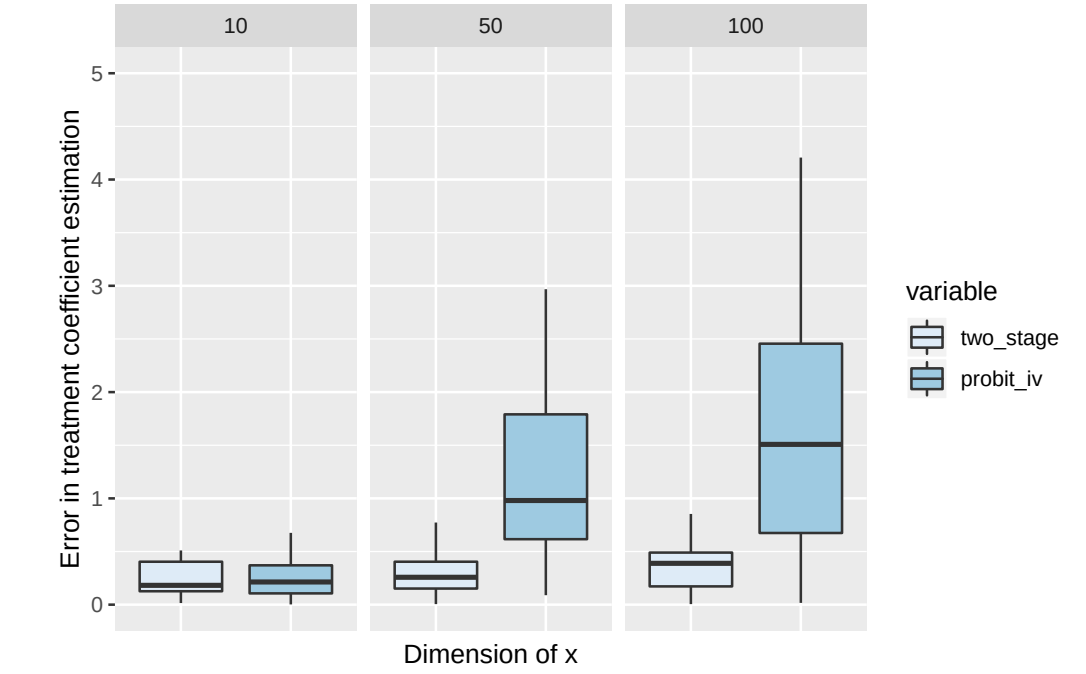
We also study a case with no endogeniety or instrumental variables ($\rho = 0,\ Z = 0$) and show that we are still able to achieve an improvement relative to traditional single stage logistic regression and logistic regression with elastic-net penalty (Zou and Hastie (2005), implemented using the `Glmnet` package (Friedman et al. (2009)), with 10 fold cross-validation). In figure 2-1b, the logistic regression is able to achieve relatively high accuracy when the number of samples n is large relative to d (the dimension of X), but performs poorly as d grows. As expected, the elastic-net logistic regression is less sensitive to the dimension of the data, but has a bias in the treatment coefficient's estimation across all datasets, which results in poor performance relative to the two stage estimator. This highlights that the benefits of the two stage approach extend beyond the scenario with endogeneity, and that an orthogonalization step is useful here as well.

2.4.2 Heterogeneous treatment effect: gradient boosted trees

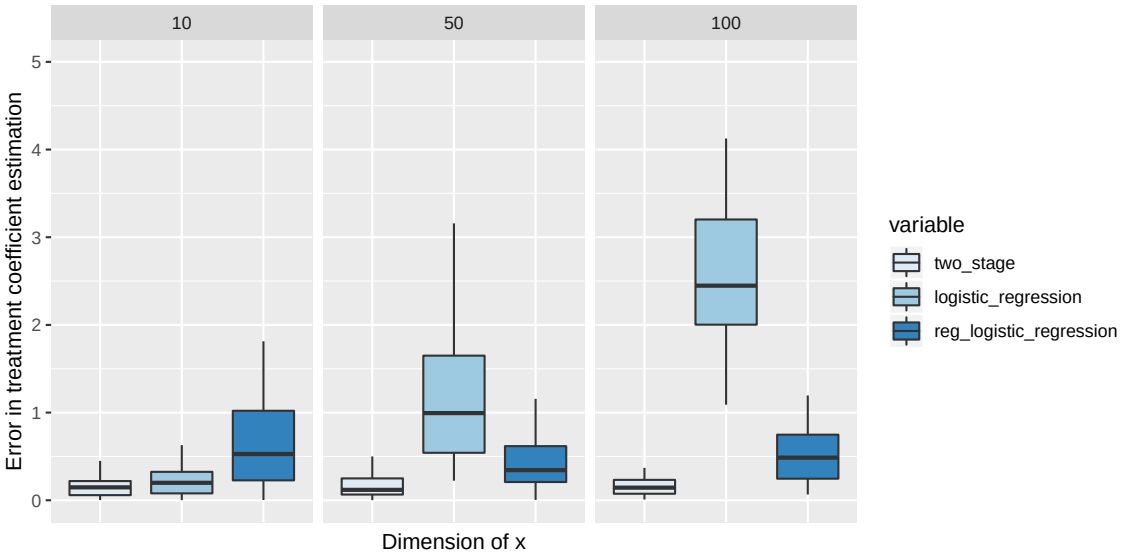
We now move to the more general case where the treatment effect depends on the covariates X. We use the model (2.1) - (2.5), with $u \sim N(0,1),\ X_i \sim N(0, I_d),\ h(x) = x_1,\ \beta = 1,\ \rho = 0.5,\ \sigma_v = 1$.

We explore a number of different synthetic datasets:

- Dataset 1: simple variable treatment effect ($d = 2$), $g(x) = x_1 + x_2,\ m(x) = x_1,\ \tau(x) = x_1$.



(a) With endogeneity



(b) Without endogeneity

Figure 2-1: Homogeneous treatment effect estimate accuracy ($n = 1000$)

- Dataset 2: linear relationships ($d = 10$), $g(x) = x_1 - x_2 + x_3$, $m(x) = x_1 + x_2 + x_3$, $\tau(x) = x_1 - x_2 + x_3$.

- Dataset 3: non-linear $g(x)$, linear $m(x), \tau(x)$ ($d = 2$), $g(x) = \sin(10(x_1 + x_2)^2))$, $m(x) = x_1$, $\tau(x) = x_1$.

- Dataset 4: non-linear relationships ($d = 2$), $g(x) = \sin(10(x_1 + x_2)^2))$, $m(x) = (x_1 - x_2)^2$, $\tau(x) = \cos(x_1 + x_2)$.

We test our different methods on the ability to correctly predict the change in probability associated with a change in the treatment. To achieve this, we change each treatment by a specified amount and recompute the predicted probability holding the control features and instrument (X_i, Z_i) constant. We then calculate the difference relative to the predicted probability of selling for the initial treatment assigned in the training data. A randomly generated sample of 1,000,000 data points was used.

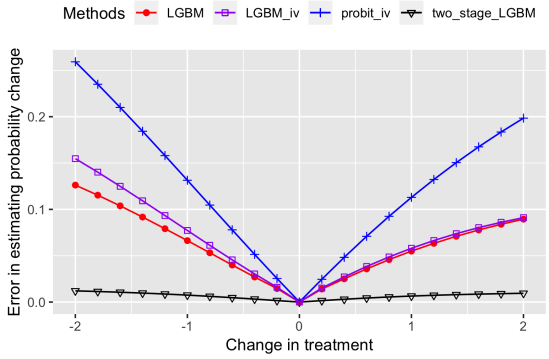
In addition to comparing against `probit_iv` which is limited by its linear form, we also compare against Blundell and Powell (2003). This provides an extension of Rivers and Vuong (1988) to the case where $g(X)$ and $h(X)$ are non-parametric functions, by replacing the steps in the procedure by the non-parametric equivalent (`LGBM_iv`):

1. Estimate $\hat{q}(X, Z) = \hat{m}(x) + \hat{\beta}Z$ as a non-parametric function. Find $\hat{\upsilon} = T - \hat{q}(X, Z)$, and $\sigma_\upsilon = \frac{||\hat{\upsilon}||}{\sqrt{n}}$.

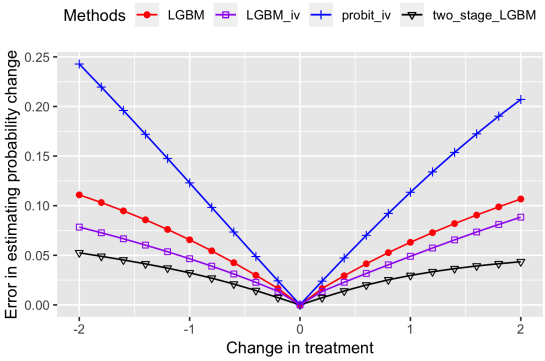
2. Estimate $\hat{f}(X, Z, T, \hat{\upsilon}) = g(X) + \tau(X)T + \frac{\rho}{\sigma_\upsilon}\upsilon$ as a non-parametric function.

We note that the main difference with (`two_stage_LGBM`) is that the `LGBM_iv` approach above does not have an orthogonalization step, and does not identify the treatment effect $\tau(X)$, although, as in section §2.4.2, this can be approximated by changing T in the function $\hat{f}(X, Z, T, \upsilon)$. Finally we compare against a `lightgbm` classifier with X, T as explanatory variables, a method denoted `LGBM`. With the `two_stage_LGBM` algorithm, we do the first and second steps using gradient boosted trees in the `lightgbm` package. 50 rounds of boosting were used for all procedures.

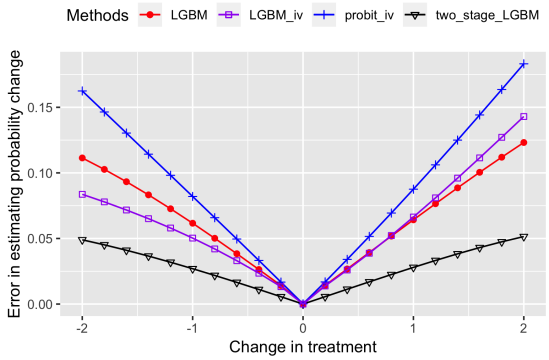
Figure 2-2 shows that `two_stage_LGBM` performs well over a range of synthetic datasets with endogentiety as long as a suitable instrument is available. The `probit_iv` method performs poorly on all models where the nuisance parameters are non-linear or the treatment effect is heterogeneous. The orthogonalization step appears to give `two_stage_LGBM` an advantage over `LGBM_iv` in almost all cases. `LGBM` is generally inferior to `LGBM_iv` due to its inability to leverage the instrumental variable to remove the effects of endogeneity.



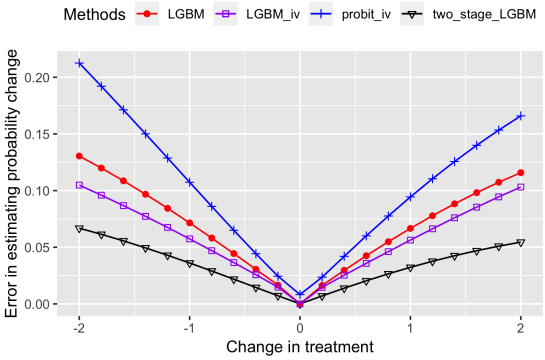
(a) Dataset 1: simple variable effect treatment



(b) Dataset 2: linear relationships



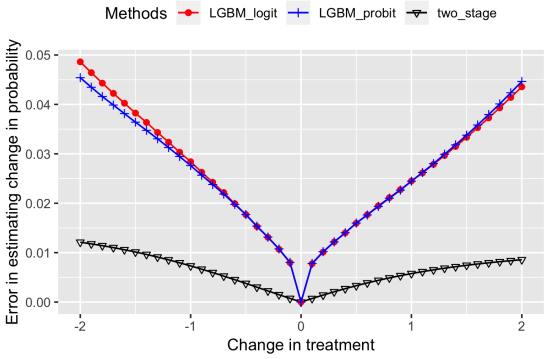
(c) Dataset 3: non-linear $g(x)$, linear $m(x), \tau(x)$



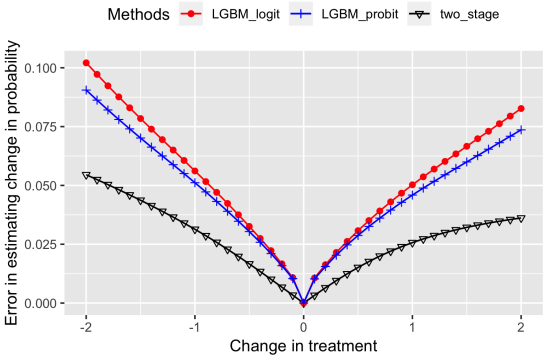
(d) Dataset 4: non-linear relationships

Figure 2-2: Estimating treatment effect for changing sample size

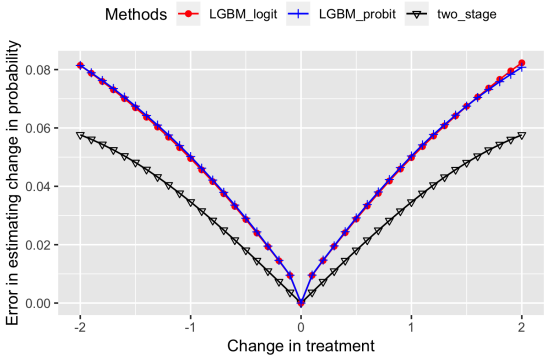
We also test our approach when there is no endogeneity ($\rho = 0$) and compare to `LGBM_logit` and `LGBM_probit` using boosted trees with a logistic or probit loss function respectively. Due to the lack of existing methods on estimating heterogeneous treatment effects in the classification setting with continuous treatments, we also tried some state-of-the-art methods from the literature meant for the regression setting,



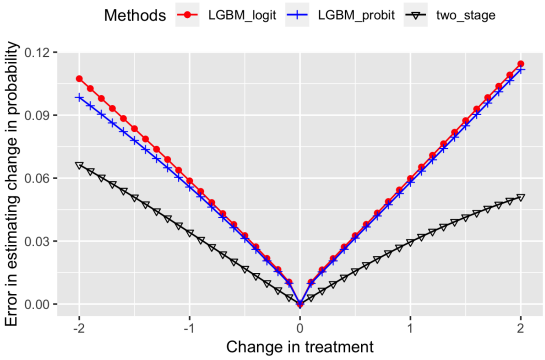
(a) Dataset 1: simple variable effect treatment



(b) Dataset 2: linear relationships



(c) Dataset 3: non-linear $g(x)$, linear $m(x), \tau(x)$



(d) Dataset 4: non-linear relationships

Figure 2-3: Estimating change in probability with a change in treatment

such as generalized random forests (Athey et al. (2016)) and `rlearner` (Nie and Wager (2017)). These methods were not insightful in the classification setting so are omitted from the following results. The results are shown in Figure 2-3. We observe that our two stage approach is able to consistently better estimate the treatment effect than single stage gradient boosted trees across all data sets in this setting.

We have additional numerical experiments in Appendix A.6.1 and Appendix A.6.2. We also explore the effect of sample size on the ability to predict the change in probability in Appendix A.6.1. In Appendix A.6.2, we explore how the treatment effect estimation impacts the (sub)optimality of the solution in synthetic pricing simulations.

2.5 Arbitrage opportunity identification

In this section, we examine two pricing models for choosing which price to sell an individual ticket, a static pricing model, where prices are held constant over the time horizon and a time-variant pricing model, where prices change over time. We then expand this to a trading algorithm and show how these pricing models can be embedded in a master problem of deciding which tickets to buy given estimated potential revenues. We address some of the assumptions and modeling choices that support this approach in section §2.5.4

2.5.1 Static pricing

To find the optimal revenue for a single ticket, the retailer optimizes the expected revenue of a ticket by choosing the optimal price $T^* = \arg\max_T \mathbb{E}[R|T, X, Z]$ where X are features of the ticket and Z is the related instrument. We denote $Y = 1$ the event that the ticket sells, then $\mathbb{E}[R|T, X, Z] = T \cdot P(Y = 1|T, X, Z)$ where $P(Y = 1|T, X, Z)$ is the probability that the ticket would sell at price T. Under the two stage classification model, we can rearrange the expression (2.9) and substitute for the maximum likelihood estimator of $P(Y = 1|T, X, Z)$. We then find an estimate

of the maximum expected revenue $\hat{R}^*(X, Z)$ by solving the following problem:

$$\hat{R}^*(X,Z) = \max_T \; T \cdot \Phi\Big(\frac{\Phi^{-1}(\hat{r}(X))k(X,\hat{\tau},\hat{\rho},\hat{w})}{\sqrt{1-\hat{\rho}^2}} + \frac{(\hat{\tau}(X)+\frac{\hat{\rho}}{\hat{\sigma_v}})(T-\hat{q}(X)) - \frac{\rho}{\hat{\sigma_v}}(Z-\hat{h}(X))}{\sqrt{1-\hat{\rho}^2}}\Big) \tag{2.17}$$

where $k(X, \tau, \rho, w)$ has been defined in Eq. (2.10). The following theorem describes the reseller's optimal pricing strategy.

Theorem 2 (Optimal Price for the reseller)**.** *For a ticket defined by covariates X:*

1. *If $\hat{\tau}(X) + \frac{\hat{\rho}}{\hat{\sigma_v}} \geq 0$, then the revenue is increasing with price and the reseller should choose the highest price possible for the ticket.*

2. *If $\hat{\tau}(X) + \frac{\hat{\rho}}{\hat{\sigma_v}} < 0$, then the revenue is unimodal, the optimal price T^* is unique and we have:*

$$T \leq \bar{T} = \frac{\frac{\Phi^{-1}(\hat{r}(X))k(X,\hat{\tau},\hat{\rho},\hat{w}) - \hat{\tau}(X)\hat{q}(X) - \frac{\hat{\rho}}{\hat{\sigma_v}}(\hat{m}(X)+\hat{\beta}Z)}{\sqrt{1-\hat{\rho}^2}}}{-2(\hat{\tau}(X)+\frac{\hat{\rho}}{\hat{\sigma_v}})} + \frac{\sqrt{(\frac{\Phi^{-1}(\hat{r}(X))k(X,\hat{\tau},\hat{\rho},\hat{w}) - \hat{\tau}(X)\hat{q}(X) - \frac{\hat{\rho}}{\hat{\sigma_v}}(\hat{m}(X)+\hat{\beta}Z)}{\sqrt{1-\hat{\rho}^2}})^2 + 8}}{-2(\hat{\tau}(X)+\frac{\hat{\rho}}{\hat{\sigma_v}})}$$

The proof can be found in Appendix A.3. In practice, in the pricing setting, the demand or likelihood of selling (a ticket in this setting) is negatively correlated with price, which means that $\hat{\tau}(X) < 0$, and furthermore our experiments suggest ρ is negative for the ticket reselling setting so the condition $\hat{\tau}(X) + \frac{\hat{\rho}}{\hat{\sigma_v}} < 0$ is always satisfied in practice. There are advantages to having the revenue be a unimodal function, it can be optimized efficiently using a line search algorithm such as Brent's method (Brent (1971)). In addition, having an upper bound helps limit the search space. Furthermore, our extensive numerical experiments showed the following results on the impact of the magnitude of price sensitivity $\hat{\tau}(X)$ on optimal price and revenue. Figure 2-4 shows that the optimal revenue and price under this model are quite robust to errors in $\tau(X)$ when the magnitude of $\hat{\tau}(X)$ is very negative, but are quite sensitive

to errors of $\hat{\tau}(X)$ if $\hat{\tau}(X)$ is small in magnitude (note since treatment effect is always less than zero in our setting, $\hat{\tau}(X) = 0$ is on the right of the plot).

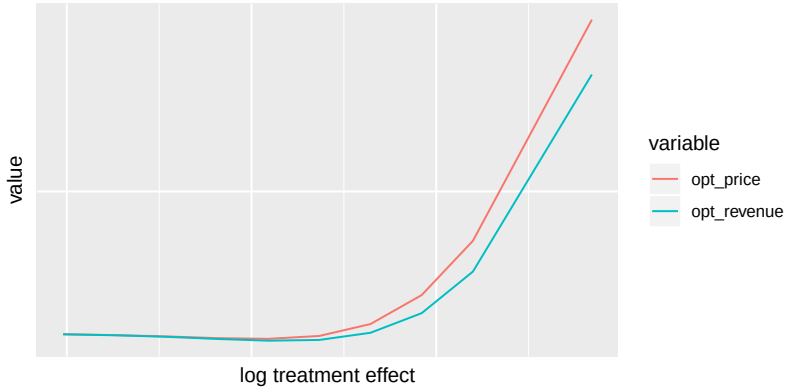


Figure 2-4: Sensitivity of optimal price and revenue to magnitude of treatment effect

2.5.2 Time-varying pricing

It is also possible to incorporate the two stage treatment estimation procedure model into a dynamic pricing model. Since our collaborators do not aim to transact large volumes, we develop a dynamic pricing model for pricing with a single ticket. If tickets being sold are small in number and diverse, the cross effects of tickets in the portfolio are small and the relative effects on the overall market are minor. We evaluate the strength of these cross-effects on the data in Appendix A.5. We further assume that demand is independent over time periods, but allow it to change over time according to a forecast. While simple, this model is sufficiently rich to capture the time dynamics of perishable inventory, and how prices fall closer to the event time to improve the chances of selling in the scarce time remaining. Furthermore, this simplicity is necessary to solve the model tractably on the scale of the millions of tickets that could possibly be purchased and sold on the market. It is desirable to be able to make these calculations in a fraction second to be able to offer the seller a price before they list their ticket.

The dynamic pricing problem for a single ticket can be formulated as a markov decision problem over a finite horizon H. Define a variable W_t to be the probability

that the ticket will sell in a given period t. The ticket features are explicitly split into M_t the market features which vary with time (discussed further in section §2.6.1) and X the ticket features which do not change over time. Note, for notational simplicity we suppress the dependence on the ticket i, but there is a unique X and M_t for each ticket. Then the Bellman equation is:

$$J_t(M_t) = \max_{T_t} T_t \cdot P(W_t = 1|T_t, X, M_t) + (1 - P(W_t = 1|T_t, X))\mathbb{E}[J_{t-1}(M_{t-1})] \tag{2.18}$$

Where the first term is the revenue achieved if the ticket sells while the second term is the expected revenue from the future. It is possible to estimate the probability of selling in a single period $P(W_t = 1|T_t, X, M_t)$ from the probability of selling in the remaining horizon $P(Y_t = 1|T_t, X, M_t)$:

$$P(W_t = 1|T_t, X, M_t) = \frac{P(Y_t = 1|T_t, X, M_t) - P(Y_{t-1} = 1|T_{t-1}, X, M_{t-1})}{1 - P(Y_{t-1} = 1|T_{t-1}, X, M_{t-1})} \tag{2.19}$$

While theoretically solvable, the dynamic programming solution isn't tractable on this scale. A common approach in the literature is to solve this model using the deterministic equivalent LP. This LP is used by Ma et al. (2018) in the pricing setting, and by several others (Gallego and Van Ryzin (1994); Wang et al. (2018)) in the revenue management and scheduling literature.

$$\bar{R}_H(X_i) = \max \sum_{j \in \mathcal{L}} \sum_{t=1}^{H} q_{tj} T_{tj} x_{tj} \tag{2.20}$$

$$\text{s.t} \sum_{j \in \mathcal{L}} \sum_{t=1}^{H} q_{tj} x_{tj} \leq 1 \tag{2.21}$$

$$\sum_{j \in \mathcal{L}} x_{tj} \leq 1 \qquad \forall t \in \{1, ..., H\} \tag{2.22}$$

$$x_{tj} \geq 0 \qquad \forall t \in \{1, ..., H\},\ j \in \mathcal{L} \tag{2.23}$$

In this formulation, the variable x_{tj} selects a price T_{tj} and quantity $q_{tj} = P(W_t =$

$1|T_{tj}, X, \mathbb{E}[M_t])$ from a discrete ladder $j \in \mathcal{L}$ for each time period, while the objective is the expected revenue over the selling time horizon. Constraint (2.21) is an inventory constraint which requires the ticket will be at most sold in expectation, while constraint (2.22) requires no more than one price selected per period.

It is well known that the objective value for this LP is an upper bound on the optimal dynamic policy (Ma et al. (2018)). Once we find optimal prices for each ticket $T_t^* = \sum_{j \in \mathcal{L}} T_{tj} x_{tj}^*$ from the LP we can use monte carlo simulation to approximate the optimal revenue that this approach can achieve:

$$\hat{R}_H^*(X) = \mathbb{E}\left[\sum_{t=1}^{H} T_t^* \cdot W_t(T_t^*, X_i, M_t)\right]$$

Note here the expectation is taken over sample paths of the (ticket dependent) market dependent features M_t. To construct these sample paths we define it using the following evolution equation:

$$M_t = M_0 + \mu_i t + \sigma_i t \epsilon$$

Where μ_i and σ_i are parameters fitted through historical data for each ticket i, and M_0 indicates the market feature at time of listing. ϵ is a standard $(0, 1)$ normally distributed variable. Therefore, each market feature is assumed to be a Gaussian random walk.

2.5.3 Global optimization for purchasing a ticket portfolio

To decide which tickets to purchase and add to our portfolio, in what follows, we introduce the following optimization problem. In practice, this optimization problem is resolved frequently over the time horizon to respond to market changes.

$$\max_{z_t} \sum_{s \in S} \sum_{t \in A_s} (\hat{R}_t^*(X_t) - T_t^0 - b) z_i + \sum_{s \in S} \sum_{t \in I_s} \hat{R}^*(X_t) \tag{2.24}$$

$$\text{s.t.} \sum_{t \in A_s} z_t + |I_s| \leq C_s \qquad \forall s \in S \tag{2.25}$$

$$z_t \in \{0, 1\}, \tag{2.26}$$

In this formulation z_t is a binary variable deciding whether each ticket t should be purchased, A_s is the set of available tickets on the market for a particular section of an event $s \in S$, and I_s is the number of tickets purchased in previous stages but were unsold by the algorithm for that section. The first term in the objective compares the optimal expected revenue to the listed price T_t^0 of the ticket. If the expected revenue sufficiently exceeds the listing price plus a buffer b, the ticket is considered for purchase. By tuning the buffer parameter b, we limit the rate at which tickets are purchased. The second term in the objective is a re-optimization of the listing prices for tickets which have been purchased by the algorithm but are unsold, based on the updated market features and the time until event. Constraint (2.25) limits the number of tickets held in inventory for a given section. This is a knapsack problem that can be decomposed by section and can be practically solved efficiently, once the price optimization has been calculated for each ticket using either (2.24) or (2.17).

2.5.4 Business considerations

The optimization problem in section §2.5.3 also incorporates some important business constraints. These constraints reflect strategic priorities and practical implementation issues.

We assume that the trading algorithm will buy/sell small volumes relative to the secondary ticket market size. This is necessary to prevent possible endogeneity issues; if a market maker were to buy too many tickets, the underlying market mechanisms would change. Thus, it would be difficult to justify using historical data to estimate price sensitivity once a new entrant becomes a sizeable market participant. We achieve this with the tunable parameter b and hard constraint (2.25), limiting the number of tickets in inventory at any given time.

We also assume there will be no interaction between tickets within our portfolio. We quantitatively evaluate the strength of these cross-effects on the data in appendix A.5.

Environment State Features (ES)	Game Features (G)
Day of Week	Home Team
Week of Year	Away Team
Month of Year	Average 3pts Attempted (Home)
Days Listed Before Game	Average 3pts Made (Home)
	Win Percentage (Home)
Market and Pricing Features (MP)	
Price of Ticket	section of Ticket
Row of Ticket	# of Tickets in Listing
25th/50th/75th Percentile Price—	25th/50th/75th Percentile Price—
Listed in Section/Game	Sold in Section/Game
Quantity Sold in Section/Game	Quantity Listed in Section/Game
Historical Game Median Price	Total Value sold in Section/Game
Normalized Quality Score by Section/Game	

Table 2.1: NBA data features

2.6 Implementation

We focus on the NBA (National Basketball Association) as a representative market for ticket reselling. We focus on the 2014-15 NBA season with 1230 games and 8.5 million ticket listings.

2.6.1 Feature generation

For a particular ticket, there are three types of features that could affect whether a ticket would sell as shown in the table 2.1:

Additional features were generated but not used in the final model due to the lack of additional explanatory power. These included: historical median price sold, historical median price sold in section, historical median price listed in section, historical quantity sold in section, historical team win percentage and historical median time to sell. For some analysis on feature importance, please refer to Appendix A.4.

2.6.2 Instrumental variables

A useful instrument is how the seller has historically priced tickets relative to the market. For each ticket the seller lists, we can calculate the difference between the chosen price and the median price sold in the section the ticket is listed in. We can

take the average of this difference over time, updating as the seller lists more tickets, and use this as the instrumental variable. If the seller has not listed a ticket before, the default value is 0. This is a suitable instrument because the only way this affects the probability the ticket is sold is through the price.

The effect of the historical price deviation is significant in predicting price. Table 2.2, shows the coefficient of the instrument in the regression of ticket and market features and the instrument on price. A Wald test of significance, with and without the IV gives a F-value of 2481.8 and p-value of $< 2.2 \times 10^{-16}$. Typically the F-value for the joint significance of the instruments should be greater than 10 (Stock and Watson (2015)). We also conducted a likelihood ratio test, obtaining an F-value of 2477.802 and p-value of $< 2.2 \times 10^{-16}$.

Table 2.2: Regression of ticket and market features and the average historical price difference on price

Variable	Estimate
Average historical price difference	0.009499***
	(0.0001907)
Observations	698286
R^2	0.6508
Adjusted R^2	0.6508
Note:	*p<0.1; **p<0.05; ***p<0.01

Table 2.3: Probit regression of price, ticket and market features and price residual (from IV model) on selling outcomes

Variable	Estimate
Price residual	0.002902***
	(0.0004329)
Observations	698286
AUC	0.8005
Misclassifcation rate	0.1942
Note:	*p<0.1; **p<0.05; ***p<0.01

It is also possible to show from the data that endogeneity exists in our model,

therefore justifying the use of an instrumental variable approach. We use the test from Guilkey et al. (1992), which incorporates the residual from the price prediction into the probit estimation for likelihood of each ticket selling. Table 2.3 shows strong evidence against the coefficient associated with the residual being zero (p-value= 2.04×10^{-11}), suggesting that there is endogeneity in the model.

2.6.3 Predictive accuracy

In table A.1, we observe the predictive power of the models introduced in section §2.4 on the NBA dataset for 2014-2015. [2] The results shown are for a testing set with a 70 - 30 split. We observe comparable performance in predictive accuracy across all gradient boosted tree methods, with regards to misclassification error and AUC. The corresponding ROC curve is shown in Figure 2-5. The two-stage LGBM method achieved the highest accuracy. We note that parametric models which do not capture the interaction terms or heterogeneous treatment effects, such as IV probit, are considerably less accurate and are possibly misspecified. This is particularly relevant as previous econometric studies ticket selling have used such models (Sweeting (2012)), which may lead to erroneous treatment effect estimation. Overall, this high predictive accuracy is promising in terms of being able to successfully trade tickets. We also show the performance on a more simple model with fewer features in Appendix A.4.

Algorithm	Misclassification error	AUC	Miscalibration
Two stage LGBM	0.135	0.908	0.0112
One stage LGBM	0.144	0.898	0.0251
Probit IV	0.190	0.795	0.0426
LGBM IV	0.142	0.901	0.0141

Table 2.4: Predictive accuracy and miscalibration on NBA data

[2] There are a number of parameters which are required to be estimated for the method. The estimated parameters are: $\hat{\sigma_u} = 58.6$, $\hat{\rho} = -0.32$ and $\hat{\sigma_v} = 0.38$. The small size of $\hat{\sigma_v}$, is explained by the fact that better results are obtained if $log(price)$ is used.

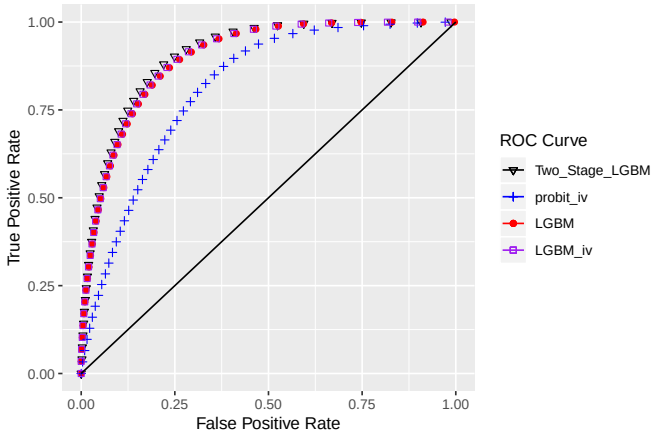


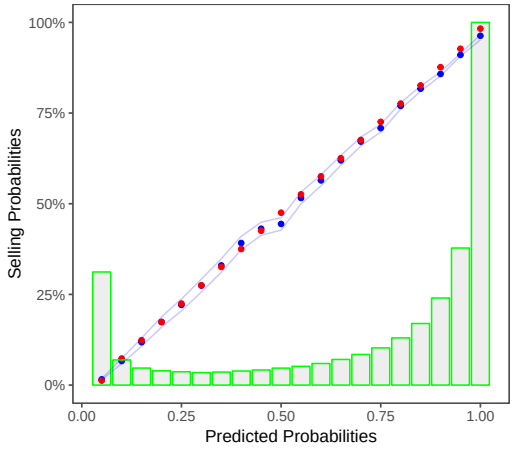
Figure 2-5: Performance on NBA data with IV: ROC curve

2.6.4 Probability calibration

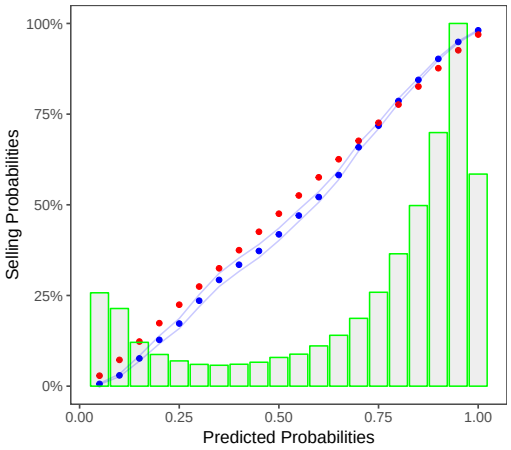
Our algorithm uses the predicted probability of a ticket selling to set the optimal price for a ticket. Therefore, it is very important to be able to estimate these probabilities accurately, which is different from just having a low misclassification rate. For example, it is a well-documented fact that gradient boosted trees do not produce well calibrated probabilities (Caruana and Niculescu-Mizil (2006)).

Figure 2-6 shows a calibration plot, which explores the difference between the predicted and empirical probabilities of the model. Empirical probabilities are obtained by binning tickets with similar predicted probabilities, and calculating the empirical selling percentage. For example, for the IV probit model, tickets with a predicted probability of selling between 22.5-27.5% actually were sold, on average, 15% of the time (as seen in the trend line with red dots in Figure 2-6c). For reference, we include an oracle which has perfect calibration as well a histogram of how many tickets are in each bin. Deviation of the red dots from the blue dots indicates miscalibration. In Figure 2-6, we observe that the two-stage approximation is significantly closer to an oracle classifier than the other models, meaning that it is able to more accurately predict the probability of selling for each ticket.

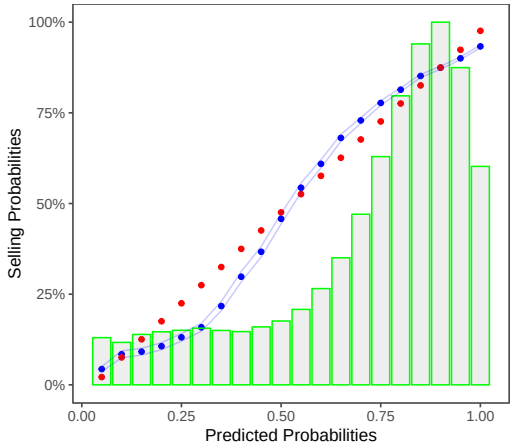
To further quantify the effect of miscalibration, we devise the following statistic. Consider a set of tickets $\mathcal{T} = \{t_i\}_{i=1\cdots n}$, and predicted probabilities $\hat{p}_i$ for ticket t_i.



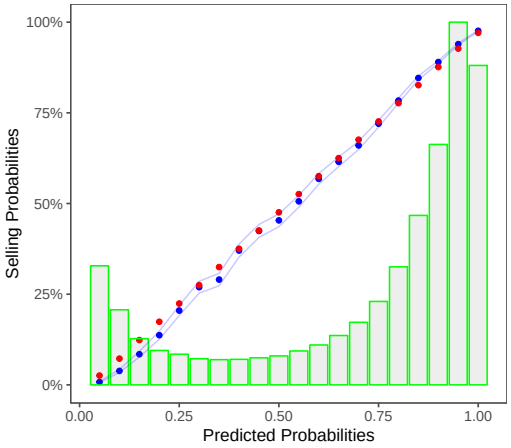
(a) Calibration curve of Two-stage Method



(b) Calibration curve of One-stage LGBM



(c) Calibration curve of the IV probit



(d) Calibration curve of LGBM IV

Figure 2-6: Calibration curves on NBA data

We create bins $M_j = [l_j, u_j)$ that separate the predicted probabilities into equally spaced intervals. Then the empirical probability of bin M_j is $p_j = \frac{1}{|i:\hat{p}_i \in M_j|} \sum_{i:\hat{p}_i \in M_j} Y_i$ where Y_i is the observed outcome for ticket t_i. We define the predicted probability of bin M_j, as $\bar{p}_j = \frac{1}{|i:\hat{p}_i \in M_j|} \sum_{i:\hat{p}_i \in M_j} \hat{p}_i$. Thus, miscalibration is defined as $\frac{1}{n} \sum_j |i : \hat{p}_i \in M_j| \cdot |\bar{p}_j - p_j|$. This can be seen as a weighted-average difference between the average model probability and the true (empirical) probabilities. In table A.1, the miscalibration of the two-stage method is significantly less than that of the other methods at just over 1%.

Optimization Model Method	Pricing Strategy	Total Profit	% of Bought Tickets Sold	ROI
Two-Stage LGBM	Constant	14.2K	61%	7%
	Time varying	18.9K	67%	11%
IV Probit	Constant	-250K	21%	-71%
	Time varying	-120K	35%	-32%

Table 2.5: Comparison of backtesting profit between methods with LGBM IV baseline (limited to purchasing 50 tickets a day)

2.6.5 Back-testing optimization

To understand the performance of our algorithm on real-life data without immediately deploying it to production, we conducted backtesting experiments to see how different models perform over past periods in a trading environment. The backtesting framework uses two models: the optimization model which is used to buy/price tickets and a baseline model which is used to simulate whether the tickets will sell over the time horizon the event date. Additional technical details about this backtesting procedure are given in Appendix A.7.

We utilize the backtesting framework as detailed in on the 2014-2015 season of NBA. The baseline model is the LGBM IV model from Blundell and Powell (2003) which is shown to have high accuracy and is well calibrated in section §2.6.3 and §2.6.4. We divide the datasets into two equal-sized sets, with the training set chronologically before the testing set. We train the baseline model on the testing dataset while we evaluate our models on the training dataset, assuming the baseline to be the ground truth. We restrict the trading algorithm to only purchase 50 tickets per day, due to the computational burden of the dynamic pricing optimization and simulation of selling probabilities, but foresee that this could be scaled significantly with parallelization. The results are shown in Table 2.5:

Table 2.5 shows that the two-stage gradient boosted tree (two-stage LGBM) performed significantly better than the IV probit model, a specification previously used in the ticket resale market. A dynamic pricing strategy based on IV probit probabilities looses 32% of capital invested, while the two stage LGBM approaches achieves a positive 11% return on investment, as evaluated by an independent baseline model.

The IV probit underestimates the price sensitivity and as a result sells just 32% of the tickets it purchases, while the two-stage LGBM method sells 67% of the tickets. Furthermore, we observe that this misestimation is worse for high value tickets, which can have a significant impact on ROI if they don't sell. This highlights that in an industry with slim margins, even slight misspecification of the demand model can result in significant losses. We observe a small but significant improvement from 7% to 11% ROI when using a time-varying pricing policy relative to a constant policy, suggesting there is improvement to be made by considering the perishable inventory effects of tickets.

2.7 Conclusions

Despite the availability of very large datasets on behaviour of buyers and sellers, estimating causal relationship between price and the probability of selling for tickets on the resale market is a challenging task. We introduced a novel loss function for heterogeneous treatment effect estimation with binary outcomes which can be easily incorporated into off-the-shelf machine learning algorithms, including neural networks and gradient boosted trees. We proved that this loss function is consistent under mild assumptions on the functional form of the true treatment effect, and prove the rate convergence. In our numerical experiments on a wide range of synthetic data sets, the two stage approach is able to consistently outperform the single stage counterpart and methods from causal inference in terms of estimation of treatment effect. On historical data, we show that our two stage approach is price sensitive with potential to earn up to an 11% return by trading on tickets on the platform. This is significant as previous IV probit methods for estimating price sensitivity in this setting are not profitable. We hope that these results increase awareness of the importance of accurately estimating the casual relationship between price and sales in the revenue management and pricing community.

The results for heterogeneous treatment effect estimation for classification are widely applicable beyond the scope of ticket reselling. We believe they could be useful

in areas such as medicine, revenue management and other econometric applications.

A sign of the ultimate success of this project is the fact that our industry collaborator is now proceeding to pilot this algorithm in real time, and we are excited to follow up the collaboration and see the pilot results very soon.

Chapter 3

Optimizing objective functions determined from random forests

3.1 Introduction

Many optimization problems arise from practitioners wanting to improve their performance by leveraging data. This data may include previous decisions made in a variety of situations, and the outcomes that result from these decisions. It is natural for practitioners to want to learn from this data, and make new decisions which are more likely to result in favorable outcomes.

As an example we will return to throughout this chapter, suppose a property investor is interested in constructing a new house with the goal to sell for a profit. The property investor needs to make several decisions such as the size of the house, number of bedrooms and bathrooms, quality of materials used, and where it will be located. Suppose the property investor has data on house sales for a region and information about the characteristics of each house sold. The investor might use this data to inform their decision about which lots to buy and what kind of houses to construct with the goal of maximizing the sales price, subject to constraints on their budget and which vacant lots are available to build a house. We formulate this optimization problem more precisely and provide insights in section 3.7.

There are many other examples of optimization problems where this type of data

may exist, some recent applications are included in the following table:

Problem	**Actions(x)**	**Scenarios/ additional data (θ)**	**Outcome(y)**
Insurance Besbes et al. (2010)	insurance rate to offer	characteristics of customer, characteristics loan requested	profit
Online retail Ferreira et al. (2015)	price of an item	characteristics of item, prices of other items	revenue
Drug trials Bertsimas et al. (2016)	chemotherapy regimens	characteristics of patients	survival/ toxicity
Promotion vehicle scheduling Baardman et al. (2015)	promotions (fliers, commercials etc) to schedule	planned prices	revenue

Table 3.1: Examples of optimization problems with an uncertain relationship between actions and outcomes

These optimization problems are generally difficult because we do not know the exact relationship between the decisions made and the outcomes which are observed, so the objective function is not well defined. This can be due to a lack of knowledge of how the underlying system works, or due to inherent uncertainty in the problem. Returning to our previous example, if the property investor decides to add an extra bathroom, it is not clear what the impact will be on the sales price of the house. Furthermore, it is not even clear if this relationship will be linear, or if there will be some interaction with other decisions (i.e does adding an extra bathroom to a large house increase price more than adding an extra bathroom to a small house?)

Fortunately, recent advances in machine learning have enabled the estimation of sophisticated relationships from data with increasing accuracy. In particular, suppose we have some data $\mathcal{D} = \{x^{(i)}, y^{(i)} | i = 1, ..., n\}$, where $x^{(i)} \sim X$ are previous actions and $y^{(i)} \sim Y$ are corresponding outcomes. It is possible to fit a machine learning function $R(\cdot)$ to our data, such that the function predicts the expected outcome for a given decision $R(x) \approx E[Y|X = x]$. Using machine learning terminology, x are known as *features*, while y are the *labels*. In the property investment example, $R(x)$

would be the function which predicts the sales price y, while x includes the size of the house, number of bedrooms and bathrooms, quality of materials used, and where it will be located.

Once we have a function relating decisions to outcomes, we can optimize over that function $R(x)$ as an objective, as a proxy for optimizing $E[Y|X = x]$ (optimizing the outcome). We can restrict the decision variables x to lie within some polyhedral or discrete set $\mathcal{X}$ to give us the following optimization problem that we solve in this chapter: $\underset{x \in \mathcal{X}}{\text{maximize}} \quad R(x)$. In general, polyhedral sets provide a very rich modeling framework, particularly when combined with integer variables. An example of a polyhedral set from the housing example could be related to ensuring the construction costs are within a budget. A very simple constraint (for illustrative purposes) might be $\textbf{cost per squarefoot} \times \textbf{house size} + \textbf{cost per batroom} \times \textbf{number of bathroms} \leq \textbf{budget}$.

In the examples we explore, the action taken is often dependent on data θ which defines the scenario under which the action was taken. We will denote this dependence by $x \in \mathcal{X}(\theta)$, where the scenario defines the set of possible actions. As an example, the lot size of the property the investor decides to build on, is a function of the set of lots which are available. [1] [2]

We will focus our attention on the case where $R(x)$ is a random forest (Breiman (2001)). A random forest is an example of an ensemble method, which combines predictions from several predictive algorithms, in our case decision trees. A decision tree uses a tree structure to predict an outcome for a feature vector x. An example of a decision tree for classifying gender is given in figure 3-2, in this case $x = (height, weight)$. At each internal node of the tree, starting at the root node, a feature is tested against a condition, for example, "is $height < 180cm$?". Depending

[1] In particular if we define $\mathcal{A}$ to be the set of available lots, $\theta_{lotsize}$ as the size of each available lot and $z_i = 1$ if lot i is chosen, 0 otherwise, we can model this as $x_{lotsize} = \sum_{i \in \mathcal{A}} \theta^i_{lotsize} z_i$, $\sum_{i \in \mathcal{A}} z_i = 1, z_i \in \{0,1\}^{|\mathcal{A}|}$.

[2] Furthermore, under this framework, it is possible that x includes features which are not under the control of the decision maker, but are predictive of the outcome. For example, suppose the current mortgage rates are predictive of the sales price. In this case, we may have a constraint such as $x_{mort} = \theta_{mort}$, so that the current mortgage rates are a feature in the prediction of sale price.

on whether this statement is true or false, the next node visited will be either be the left or the right child node. After being evaluated on a sequence of these conditions, the feature will be assigned to a leaf node which is associated with a prediction. For example, if the height of the person being classified is less then 180cm, next we would check whether their weight is greater than 80kg. If this is true, we would predict this person is a man, otherwise we would predict they are a woman.

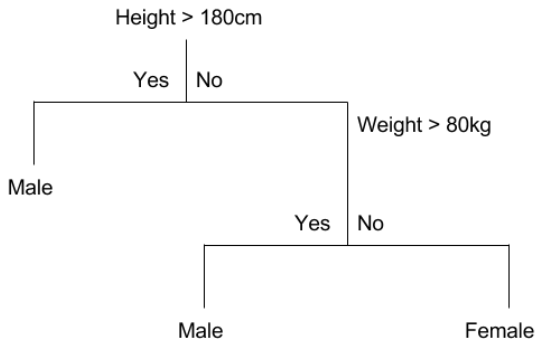


Figure 3-1: Example tree for classifying gender

In practice, individual decision trees are often associated with high variance predictions and poor out-of-sample predictive accuracy. They will often overfit the data if they are not restricted in size (Breiman (1996)). Random forests decrease this variance by bagging, which involves taking the average of many trees, each trained by randomly sampling the training data and features that are used in the splits (Breiman (2001)). Random forests (and indeed our methodology) can be applied for either regression or classification.

Using a random forest as an objective function has many advantages. Random forests are a very powerful prediction algorithm which can fit unknown non-linear and complex interactions of features with minimal feature engineering, making them popular and widely used in machine learning (Biau and Scornet (2016)). Random forests often perform very well in practice, with high prediction accuracy over a wide range of applications from 3D image recognition (Shotton et al. (2013)) to econometrics (Varian (2014)). Furthermore, they are generally recognized as being able to deal ef-

fectively with small sample sizes and high dimensional data, and have few parameters to tune (Biau and Scornet (2016)). Of particular relevance to optimization, random forests have special combinatorial structure which allows us to leverage the power of MILP and powerful existing solvers to solve large scale problems. The methodology in this chapter can easily be extended to other tree based ensemble methods, including XGBoost (Chen and Guestrin (2016)) and AdaBoost (Freund and Schapire (1995)), which also have the same structure as random forests.

Underlying this chapter is the assumption that the random forest estimated from the data is a good approximation of the underlying relationship between the actions and the outcomes. This is not always true, as there are potential issues with confounding and non-identifiability as has been documented in Bertsimas and Kallus (2016b). However, we note that this is practically a difficult issue to address, and believe these models can still offer insight, as long as the practitioner is aware of the potential limitations.

Contributions: First, we show how to *solve optimization problems with random forest objective functions, with general polyhedral constraints*. We show that it is possible to formulate this problem as a MILP and show this formulation can be decomposed and solved iteratively using Pareto-optimal Benders cuts. Although the subproblems contain complicated binary logic, we provide a naturally integer subproblem formulation. This in turn allows us to consider the dual formulation and hence consider Benders cuts.

Second, *we provide analytical guarantees on an approach that approximates a large scale random forest optimization problem by optimizing over a smaller forest.* We show that the sub-optimality of the approximate solution decays exponentially with the number of trees in the smaller random forest, under certain assumptions on how trees are generated.

Third, for very large problem instances, *we propose heuristic algorithms which optimize over smaller forests using an approach inspired by cross-validation.* We derive analytical upper and lower bounds on the performance of the optimal solution from the heuristic.

Finally, *we explore the performance of these algorithms using two original case studies with real data; a property investment and a jury selection case study.* We show that our approaches outperform benchmarks optimizing other machine learning objective functions.

3.2 Relevant literature

There are many different algorithms for generating decision trees such as ID3 (Quinlan (1986)), C4.5 (Salzberg (1994)) and CART (Breiman et al. (1984)), which Breiman uses in the original random forest paper. Trees can be used for either classification or regression. The general procedure is to greedily partition the data along axis aligned splits, to maximize homogeneity of the resultant sets as evaluated by a metric. For classification trees, CART uses Gini impurity as the metric, while C3 and ID4.5 use information gain. For regression, variance reduction is used. We also note the splits can be oblique (defined by a hyperplane - see Murthy et al. (1994)), which our method is sufficiently general to capture. The prediction for a leaf is the average of the training labels for regression and the most common class in classification. There have been recent efforts to apply modern optimization techniques to find globally optimal classification trees, rather than following a greedy heuristic as has been done in CART. Bertsimas and Dunn (2017) present a mixed-integer linear programming (MILP) formulation and show it outperforms widely used heuristics for datasets of a reasonable size. They focus on training trees (optimal estimation), whereas our optimization is focused on choosing the best 'input feature vector' *given* an ensemble of trees. As such our MILP formulation, decomposition and heuristics are different.

Despite significant academic interest, there has been limited success in developing theory to explain the success of random forests. In a recent review Biau and Scornet (2016), Biau and Scornet observe "*On the theoretical side, the story of random forests is less conclusive and despite their extensive use, little is known about the mathematical properties of the method*" and further "*most theoretical papers focus on simplifications or stylized versions of the the standard algorithm where analysis is*

more tractable". The difficulties involved include rigorously analyzing bagging, the CART split criteria and their dependence on the training set. To simplify the model, many papers ((Breiman (2004), Biau et al. (2008), Scornet (2014)) look at a *centered forest*, where there is no resampling of the data, the feature involved in each split is chosen at random and the split is made at the center of the domain of each feature (assumed to be $[0, 1]$). Under this model, consistency can be proved for both classification and regression forests, and the rates of convergence have been analyzed. In a high dimensional sparse setting, where there are few 'strong variables' which define the underlying function, Breiman (2004) and Biau (2012) show that the rate of convergence only depends on the number of strong variables and not the total dimension of the feature space. Another significant theoretical contribution was made by Strobl et al. (2008), who discovered the connection between random forests and a class of nearest neighbor predictors. There has been some recent success on Breiman's original model (without the aforementioned simplifying assumptions), beyond the original work. Breiman (2001) studied the generalization error of random forests, and produced bounds dependent on the correlation and strength of individual trees. More recently, Scornet et al. (2015) prove a consistency result in Breiman's setting when applied to additive regression models and pruned trees. Without pruned trees, they need to make unverifiable assumptions about behavior of the CART algorithm. Wager (2014) proves asymptotic normality of Breimans' original forests, under some assumptions on the split procedure and using different datasets to construct the tree and estimate the value of the leaves. Wager and Athey (2017) extends this analysis for causal inference. Our analysis avoids most of these difficulties because we do not focus on the process of estimating a random forest from data. Instead, we assume there is a *given* target forest we are trying to optimize over, and want to analyze the sub-optimality of the solution our method selects compared to the solution the target forest would select. We are not concerned with how this target random forest was formed, but are satisfied that it exists and provides structure for us to optimize over.

There are examples in the literature which use machine learning to estimate an objective function for an optimization problem from data. Perhaps the simplest ap-

proach is to use linear regression to estimate the unknown coefficients in the objective function. This results in a linear objective function in the optimization formulation. An example of this is Bertsimas et al. (2016) which uses ridge regression in a data driven approach to design chemotherapy regimens for cancer. There have also been efforts to model non-linear objective functions. Baardman et al. (2015) use multiplicative regression (log-linear) in order to estimate demand for promotion vehicles that can be subsequently, utilized to maximize revenue. Besbes et al. (2010) optimize over a Nadaraya-Watson kernel regression function, where the kernel is Gaussian. However, they simply evaluate their objective at a discrete number of points rather than globally optimizing over the entire function. Further examples of operations management problems incorporating covariate data into optimization problems include Chen et al. (2015), Ban and Keskin (2017),Cohen et al. (2016), Ban et al. (2016).

There is some recent literature which incorporates random forests into a predict then optimize framework. Ferreira et al. (2015) look at a pricing problem in retail. They have binary variables which decide whether to apply a price from a discrete ladder to a product. They use a random forest to estimate the demand for the product conditioned on the price, and are able to a priori make a prediction for all price-product combinations. This avoids having to optimize over the structure of the random forest, which would be necessary if the set of prices was significantly larger or continuous. Indeed, these predictions could have come from some other machine learning model, without changing the structure of the optimization problem.

Another developing stream of literature looks at combining the estimation and optimization. Rather than using a 'predict then optimize' approach to first maximize accuracy of the predicted parameter then solve the optimization problem, Elmachtoub and Grigas (2017) use a special loss function when estimating the parameter which incorporates features of the optimization problem. Rudin and Vahn (2014) use machine learning models to predict directly the optimal solution from data for a Newsvendor problem.

A sampling approach which incorporates machine learning is studied in Bertsimas and Kallus (2014). This builds on sample average approximation (SAA) (Kleywegt

et al. (2002)), where an uncertain parameter is sampled and those samples are used to solve an optimization problem that is optimal with respect to those samples. The improvement over SAA is that this approach leverages additional covariate information by sampling according to weights derived from machine learning functions. Ban et al. (2017) follow a similar idea by constructing demand scenarios from residuals from a regression model, applied to a multiperiod problem. Neither of these approaches exploit the structure of the machine learning function, as we do with random forests.

3.3 Outline

In section 5.2 we introduce the model and the MIP formulation. In section 3.5 we show how to decompose this MIP using Benders decomposition and implement Pareto-optimal Benders cuts. In section 3.6 we prove some analytical results about the sub-optimality of a smaller random forest as an approximation to a large forest, and introduce a heuristic based on cross-validation for optimization. In section 3.7 and 3.8 we introduce case studies on property investment and jury selection respectively, while in section 3.9 we explore properties of the random forest algorithms on these datasets. Finally, in section 3.10 we compare optimizing over a random forest objective function compared to other machine learning objective functions.

3.4 Model

As discussed in the introduction, suppose we have some data $\mathcal{D} = \{x^{(i)}, y^{(i)} | i = 1, ..., n\}$, where $x^{(i)} \sim X$ are previous decisions and $y^{(i)} \sim Y$ are corresponding outcomes. We train a random forest $R(\cdot)$ on $\mathcal{D}$, such that $R(x) \approx E[Y|X = x]$. Suppose we want to maximize $R(x)$ where x are decision variables that lie in some feasible polyhedral set $\mathcal{X}(\theta) = \{B(\theta)x \leq d(\theta)\}$, where θ is scenario dependent data. Concrete examples of data driven optimization problems with decisions, outcomes, and scenario-dependent polyhedral sets are given in sections 3.7 and 3.8.

The random forest is the average $R(x) = \frac{1}{T}\sum D_t(x)$, where $D_t(\cdot)$ is a decision

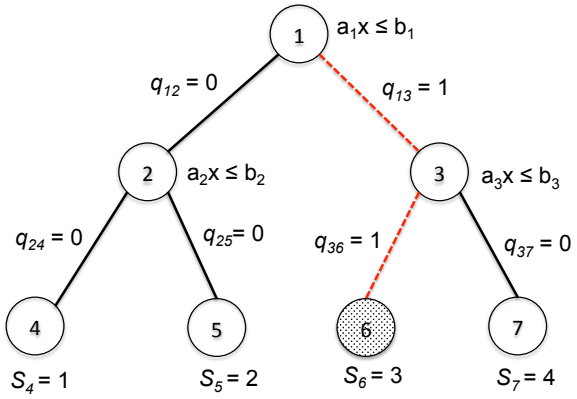


Figure 3-2: Example tree to illustrate formulation with given x belonging to leaf 6

tree and T is the number of trees in the random forest. Indeed, $D_t(x)$ is the score that a single tree t predicts for a vector x. A classification tree works by iteratively splitting the space of x, by branching on conditions relating to those features. After a number of splits, we arrive at the leaf, which is a set of x that share the same score. The branching behavior of the tree is captured with the following condition: At every interior node i, there is a linear condition on the vector x, $(a_i^t)^T x - b^t$. If $(a_i^t)^T x - b_i^t \geq 0$ then we move to the right child node of i, otherwise we move to the left child node of i. This could also be a more simple axis aligned split $x_{\sigma(i)} \leq b_i^t$, where $\sigma(i)$ is the feature index corresponding to split or node i in tree t. Note that the random forest function R (with $(a^t)_{t=1,\cdots,T}$ and $(b^t)_{t=1,\cdots,T}$) is trained from the data and is a fixed input to the optimization model.

We introduce the following notations for our model: For every tree $t \in \{1, ..T\}$, denote N^t the set of interior nodes (not leaves) in tree t. Specifically, N_r^t (respectively N_l^t) is the set of interior right nodes (respectively left nodes). Let L^t be the set of leaves in tree t. Specifically, L_r^t (respectively L_l^t) is the set of right leaves (respectively left leaves). For every node $i \in N^t$, let p_i, l_i and r_i be respectively the immediate parent, the left child and the right child of node i. S_j^t is the score of each leaf $j \in L^t$.

We introduce binary variables to track which leaf a solution x belongs to, according to the branches of the tree. For every node i and $j \in \{r_i, l_i\}$, $q_{i,j}^t$ is a binary variable such that $q_{i,j}^t = 1$ if and only if the solution x is in a leaf that is a descendant of node j. This notation is illustrated in figure 3-2 which gives an example of how

variables are set for a given tree. In view of the introduced notation, we can write $D^t(x) = \sum_{j \in L^t} S_j^t q_{p_j,j}^t$ as the score of a tree and thus our problem is to maximize:

$$\underset{x,q^t}{\text{maximize}} \quad R(x) := \frac{1}{T} \sum_{t=1}^{T} D^t(x)$$

In other words, this objective is maximizing the average of the scores of the leaves that are active because the solution x lies within the leaf. In the following, we will present a MIP formulation that describes these tree dynamics.

3.4.1 MIP Formulation

We need logical constraints which determine which leaf a solution x is in. If at an interior node i in tree t we move to the right child of i, i.e., $(a_i^t)^T x \geq b_i^t$, we force the MIP to only consider the leaves that are a descendant of right child of node i, by setting q_{i,l_i}^t to 0. Likewise, if $(a_i^t)^T x \leq b_i^t$, we set q_{i,r_i}^t to 0. These two logical constraints can be modeled using the big M-method :

$$(a_i^t)^T x - M(1 - q_{i,l_i}^t) \leq b_i^t, \qquad \text{and} \qquad (a_i^t)^T x + M(1 - q_{i,r_i}^t) \geq b_i^t.$$

This assumes that x is bounded, so that $\underline{x}_i \leq x_i \leq \overline{x}_i$. If the splits are axis aligned, as is the case for Breiman's original random forest, we can replace M with $\max\{|\underline{x}_{\sigma(i)}|, |\overline{x}_{\sigma(i)}|\} + b_i^t$. Similarly for oblique splits, we replace M by $\max_{\underline{\mathbf{x}} \leq \mathbf{x} \leq \overline{\mathbf{x}}} (a_i^t)^T x + b_i^t$.

We can now write the following MIP formulation:

$$\begin{aligned}
\underset{x,q^t}{\text{maximize}} \quad & \frac{1}{T} \sum_{t=1}^{T} \sum_{j \in L^t} S_j^t q_{p_j,j}^t \\
\text{subject to} \quad & (a_i^t)^T x - M(1 - q_{i,l_i}^t) \leq b_i^t, \quad \forall t \in \{1, \ldots, T\}, i \in N^t && \text{(3.1a)} \\
& (a_i^t)^T x + M(1 - q_{i,r_i}^t) \geq b_i^t, \quad \forall t \in \{1, \ldots, T\}, i \in N^t && \text{(3.1b)} \\
& q_{i,r_i}^t + q_{i,l_i}^t = q_{p_i,i}^t, \ \forall t \in \{1, \ldots, T\}, i \in N^t && \text{(3.1c)} \\
& \sum_{i \in L^t} q_{p_i,i}^t = 1, \quad \forall t \in \{1, \ldots, T\} && \text{(3.1d)}
\end{aligned}$$

$$q^t_{i,l_i}, q^t_{i,r_i}, q^t_{p_i,i} \in \{0,1\}, \quad \forall t \in \{1,\dots,T\}\ , i \in N^t \tag{3.1e}$$

$$B(\theta)x \leq d(\theta) \tag{3.1f}$$

Constraint (3.1c) enforces that if a parent node is not active then the children will not be active either, but if a child is active then the parent is active too. Constraint (3.1d) enforces that for each tree t, there is only one leaf that is active. Constraints (3.1a) - (3.1e) are sufficient to uniquely determine the active leaf for a given solution x for every tree. Constraint (3.1f) is the (possibly scenario dependent) problem specific feasible set for our decision

3.5 Benders decomposition

The previous formulation can be slow to solve to optimality for large problem instances, as explored in Section 3.9. Fortunately, this problem has structure which makes it an attractive candidate for Benders decomposition with lazy constraint generation. We note that this problem can be formulated as a MILP. The problem is decomposed into the master problem (MP) that chooses a solution x, and T subproblems that find the variables which determine the score of the leaf for each tree t for a given solution x to the master problem.

One requirement for Benders decomposition is that the subproblem is a linear program (i.e doesn't have integer variables). We show that if we introduce additional variables and reformulate the problem 3.1, the resulting subproblem is naturally integer. Specifically, we introduce additional binary variables to explicitly model whether the condition at each split i is met for tree t: $g^t_i = 1$ iff $(a^t_i)^T x \leq b^t_i$, 0 otherwise and formulate the master and subproblems as follows:

$$\underset{x,g^t}{\text{maximize}} \quad \frac{1}{T}\sum_{t=1}^{T} V^t(g^t) \tag{3.2a}$$

$$\text{subject to} \quad B(\theta)x \leq d(\theta) \tag{3.2b}$$

$$g_i^t \leq 1 + \frac{(a_i^t)^T x - b_i^t}{M} \leq g_i^t + 1 \quad \forall t \in \{1, \ldots, T\}, i \in N^t \tag{3.2c}$$

$$g_i^t \in \{0, 1\} \quad \forall t \in \{1, \ldots, T\}, i \in N^t \tag{3.2d}$$

$$\underset{q^t}{\text{maximize}} \quad V^t(g^t) = \sum_{j \in L^t} S_j^t q_{p_j, j}^t \tag{3.3a}$$

$$\text{subject to} \quad q_{i,l_i}^t \leq 1 - g_i^t \quad \forall i \in N^t \tag{3.3b}$$

$$q_{i,r_i}^t \leq g_i^t \quad \forall i \in N^t \tag{3.3c}$$

$$q_{i,r_i}^t + q_{i,l_i}^t = q_{p_i,i}^t, \ \forall t \in \{1, \ldots, T\}, i \in N^t \tag{3.3d}$$

$$\sum_{i \in L^t} q_{p_i,i}^t = 1, \quad \forall t \in \{1, \ldots, T\} \tag{3.3e}$$

$$0 \leq q_{i,l_i}^t, q_{i,r_i}^t, q_{p_i,i}^t \leq 1, \quad \forall t \in \{1, \ldots, T\}, i \in N^t \tag{3.3f}$$

With these new variables, each subproblem is naturally integer for a given $\{g_i^t\}_{i \in N^t}^{t=1,..,T}$. This is because each subproblem can be considered as a capacitated network flow problem, where $q_{i,j}^t$ are the flow on each node and g_i^t or $1 - g_i^t$ are the capacities, which take value 0 or 1.

Lemma 1. *Formulation (3.3) is naturally integer.*

Due to the subproblem being a linear program, we can take the dual of the subproblem:

$$\underset{r^t, l^t, v^t, u^t}{\text{minimize}} \quad \sum_{i \in N^t} (r_i^t g_i^t + l_i^t (1 - g_i^t)) + v^t - u_0^t \tag{3.4a}$$

$$\text{subject to} \quad r_{p_j}^t - u_{p_j}^t + v^t \geq S_j^t \quad \forall j \in L_r^t, \tag{3.4b}$$

$$l_{p_j}^t - u_{p_j}^t + v^t \geq S_j^t \quad \forall j \in L_l^t \tag{3.4c}$$

$$r_{p_i}^t - u_{p_i}^t + u_i^t \geq 0 \quad \forall i \in N_r^t, \tag{3.4d}$$

$$l_{p_i}^t - u_{p_i}^t + u_i^t \geq 0 \quad \forall i \in N_l^t, \tag{3.4e}$$

$$r_i^t \geq 0, l_i^t \geq 0 \quad \forall i \in N^t \tag{3.4f}$$

$$\tag{3.4g}$$

By strong duality, the optimal objective to the primal subproblem (3.3) is equal to the optimal objective of the dual (3.4). If we enumerate the extreme points of the dual solution $\mathbf{ext}(D^t)$, then (3.1) is equivalent to the following Mixed-Integer problem:

$$\underset{x,\gamma^t,g^t}{\text{maximize}} \quad \frac{1}{T}\sum_{k=1}^{T}\gamma^t \tag{3.5a}$$

$$\text{subject to} \quad B(\theta)x \leq d(\theta) \tag{3.5b}$$

$$g_i^t \leq 1 + \frac{(a_i^t)^T x - b_i^t}{M} \leq g_i^t + 1 \quad \forall t \in \{1,\dots,T\}\,, i \in N^t \tag{3.5c}$$

$$\gamma^t \leq \sum_{i\in N^t}(r_i^t g_i^t + l_i^t(1-g_i^t)) + v^t - u_0^t \tag{3.5d}$$

$$\forall t \in \{1,\dots,T\},\ \forall\ \mathbf{r^t}, \mathbf{l^t}, \mathbf{v^t}, \mathbf{u^i}\ \in\ \mathbf{ext}(\mathbf{D^t}) \tag{3.5e}$$

$$g_i^t \in \{0,1\} \quad \forall k \in \{1,\dots,T\}\,, i \in N^t \tag{3.5f}$$

We can then generate constraints iteratively using Benders cuts, rather than having a constraint for every extreme point of the dual, as outlined in Algorithm 2.

We use the standard convergence criterion for Benders decomposition, which is when the gap between the upper bound (the objective value of the most recent solution to our master problem) and the lower bound (the best previous solution as evaluated by the dual objectives) becomes less than some ϵ.

Using Benders cuts results in a significant speed improvement because we only generate cuts for the objective as required. This improvement is explored numerically in section 3.9. To generate Benders cuts, we can feed the dual linear problem (3.4) to an LP solver to retrieve the dual solution (r^t, l^t, v^t, u^i). However, thanks to the special structure of the the dual problem, we can find an explicit analytical expression of a viable solution to (3.4). This will result in significant speed since we do not need to every time solve a LP to generate a cut. We provide the expression of the analytical solution in appendix B.1.1.

An issue with our dual subproblem is degeneracy. There is often multiple optimal dual solutions. Pareto-optimal cuts have been implemented in many applications

Algorithm 2 Benders Cuts

1: *Start with* $x^{(0)} \in \mathcal{X}$:
2: *At iteration* j:
3: Solve $(D^t)^{(j)}(x^{(j)})$ using $g_i^t(x^{(j)})$ to find dual variables $(r^t)^{(j)}, (l^t)^{(j)}, (v^t)^{(j)}, (u^t)^{(j)} \quad \forall t \in \{1, \ldots, T\}$
4: Solve master problem

$$(x^{(j+1)}, (\gamma^t)^{(j+1)}, (g^t)^{(j+1)}) = \tag{3.6a}$$

$$\underset{x,\gamma^t,g^t}{\operatorname{argmax}} \frac{1}{T}\sum_{t=1}^{T}\gamma^t \tag{3.6b}$$

$$\text{s.t.} \gamma^t \leq \sum_{i \in N^t}(((r_i^t)^{(m)} - (l_i^t)^{(m)})g_i^t + (l_i^t)^{(m)}) + (v^t)^{(m)} - (u_0^t)^{(m)} \tag{3.6c}$$

$$\forall t \in \{1, \ldots, T\}, \forall m \in \{0, \ldots, j\} \tag{3.6d}$$

$$g_i^t \leq 1 + \frac{(a_i^t)^T x - b_i^t}{M} \leq 1 + g_i^t \quad \forall t \in \{1, \ldots, T\}, \forall i \in N^t \tag{3.6e}$$

$$g_i^t \in \{0, 1\} \quad \forall t \in \{1, \ldots, T\}, \forall i \in N^t \tag{3.6f}$$

$$B(\theta)x \leq d(\theta) \tag{3.6g}$$

5: *Reiterate step 2 until convergence.*

employing Benders decomposition (Mercier et al. (2005), Santoso et al. (2005)) and in the work conducted by Magnanti and Wong (1981, 1984); Tang et al. (2013), it has been shown that their use can speed up the convergence of the algorithm[3]. While Magnanti and Wong (1981) show a method to generate Pareto-optimal cuts when the subproblem dual's objective is linear with the variable vector x, in appendix B.1.2, we show in detail how we generate Pareto-optimal cuts in our particular setting where the subproblem is non-linear with x (but still linear with respect to the auxiliary variable vector g), to improve the Benders formulation.

3.6 Approximating large random forests

Suppose we have a large target forest R_T with T trees that we would use as our ideal objective function. Without loss of generality, for a finite dataset and fixed

[3]When dealing with a degenerate subproblem with multiple optimal dual solutions, the Magnanti-Wong method selects the solution that is the closest to the interior of the master problem polyhedron.

parameters, there is a finite (albeit very large) number of trees that can be formed by randomly subsetting the data and selecting the features used to construct each tree. To solve the problem in a reasonable amount of time, we can sample the trees which make up this target forest and optimize over a smaller, sampled forest. The score for a feature vector x in the target random forest $R_T(x)$ is the average of the scores of the trees evaluated at x, where $D^t(x)$ is the score of tree t, i.e., $R_T(x) = \sum_{t=1}^{T} \frac{D^t(x)}{T}$.

Alternatively, we can think of the random forest as being a partition of the feature space into disjoint sets (more specifically hyperrectangles), that we denote $\mathcal{P}$. We assume a nested structure for the partition. Specifically, if a split appears in a tree, it also appears in the random forest. This assumption is satisfied by construction of the random forest. Each tree t can also be thought of as a partition of the feature space into disjoint sets. Denote $\mathcal{P}^t$ the partition of the feature space that results from tree t. The partition of the forest $\mathcal{P}$ is formed from the union of all possible intersections of sets from the partitions of trees in the forest $(\mathcal{P}^t)_{t=1,\cdots,T}$. Namely,

$$\mathcal{P} = \bigcup_{\substack{H^{(1)} \in \mathcal{P}^1 \\ H^{(2)} \in \mathcal{P}^2 \\ \cdots \\ H^{(T)} \in \mathcal{P}^T}} \bigcap_{t=1}^{T} H^{(t)}.$$

We will label the non-null sets in the partition of the random forest $\mathcal{P}$ as $\omega_1, ..., \omega_M$. This restricts the search for the optimal feature vector to M different feature vector regions, where M is the number of sets in the partition of the target forest. We adjust the notation to the search over the finite sets ω_i. Denote S_i to be the score of set ω_i in the target random forest and X_i^t be the score of tree t for ω_i. Suppose we look at the scores of a forest R_n composed of a random sample of n trees and let $Y_i = \sum_{t=1}^{n} \frac{X_i^t}{n}$. We note that X_i^t and Y_i are random variables because they depend on which trees are sampled. Indeed, Y_i is a noisy estimate of the score S_i of the set ω_i as evaluated on the target random forest R_T[4]. The only assumption we make

[4]Although we are analyzing the scores of the sampled forest with respect to the partition induced by the target forest, we can still have an arbitrary distribution and an arbitrary dependency structure for scores X_i^t, allowing any distribution of trees to be modeled. Indeed, within a tree t, any number of scores X_i^t for adjacent hyperrectangles/sets could be equal (for trees where the partition is more

(for Theorem 3) is independence of scores between trees, i.e., for any pair of trees t_1, t_2 we have $X_i^{t_1}$ independent from $X_i^{t_2}$. This assumption is satisfied if the trees are sampled independently with replacement from the target random forest. This assumption is also satisfied when creating trees in Breiman's original algorithm as long as we condition on the training data, as the random variables defining data sampling, feature sampling and split generation are independent. We allow the scores of sets in a partition within the same tree to have an arbitrary dependence structure.

If we optimize over the sample random forest, the set which the chosen solution belongs to would be: $i^* = \arg\max_i \ Y_i \text{ s.t } \ \omega_i \cap \mathcal{X} \neq \emptyset$. This is the set with the highest score which contains a feasible solution. Let us call this sample optimal partition index i^*. i^* is also a random variable, because the set we choose in the sampled forest will depend on which trees are sampled. However, we are really interested in the score of this solution as evaluated on the target random forest, as this is our best estimate of the performance of the solution. We can quantify the sub-optimality of our solution by calculating $S_{max} - S_{i*}$, where S_{max} is the optimal objective of the target random forest: $S_{max} = \max_i \ S_i \text{ s.t } \omega_i \cap \mathcal{X} \neq \emptyset$.

We will next present a proof which shows that this optimality gap decays exponentially with the number of trees n generated from the random forest. First, we prove a lemma (required to prove the main result) which is a version of Hoeffding's inequality applied to the difference of the sample average random variables.

Lemma 2. *Let $Z_i = (X_i, W_i)$ be iid random variables each bounded by the interval $[a, b]$, with $E[X_i] = \mu \leq E[W_i] = \tau$. We have:*

$$P(\sum_{i=1}^{n} \frac{X^i}{n} \geq \sum_{i=1}^{n} \frac{W_i}{n}) \leq e^{\frac{-n}{2(a-b)^2}(\mu-\tau)^2}.$$

We leave the proof for appendix B.2. Next, we show a general result on the decay of the optimality gap only assuming independence between trees, but allowing the scores of partitions within the same tree to have an arbitrary dependence structure.

coarse than that in the target random forest).

Theorem 3. *Let* $\{X_i^t\}_{i=1,\cdots,M}^{t=1,\cdots,T}$ *be random variables, independent with respect to* t*, that lie in the interval* $[a,b]$*, with* $E[X_i^t] = S_i \ \forall t$*. Let* $Y_i = \sum_{t=1}^{n} \frac{X_i^t}{n}$ *and* $i^* = \{\arg\max_i Y_i | \omega_i \cap \mathcal{X} \neq \emptyset\}$*, while* $S_{max} = \{\max_i S_i | \omega_i \cap \mathcal{X} \neq \emptyset\}$*. Define* $\mathcal{M}(\epsilon) = \{i | S_{max} - S_i > \epsilon\}$*. Then for any* $\epsilon > 0$*,*

$$P(S_{max} - S_{i^*} > \epsilon) \leq |\mathcal{M}(\epsilon)| e^{\frac{-n}{2(b-a)^2}\epsilon^2}.$$

Proof.

$$\begin{aligned}
P(S_{max} - S_{i^*} > \epsilon) &= P(\bigcup_{j|S_{max}-S_j>\epsilon} 1_{\{j=i^*\}}) && (3.7)\\
&= P(\cup_{j|S_{max}-S_j>\epsilon} Y_j \geq \max_{k\neq j} Y_k) && (3.8)\\
&\leq \sum_{j|S_{max}-S_j>\epsilon} P(Y_j \geq \max_{j\neq k} Y_k) && (3.9)\\
&\leq \sum_{j|S_{max}-S_j>\epsilon} P(Y_j \geq Y_{\max}) && (3.10)\\
&\leq \sum_{j|S_{max}-S_j>\epsilon} e^{\frac{-n}{2(b-a)^2}(S_{max}-S_j)^2} && (3.11)\\
&\leq |\mathcal{M}(\epsilon)| e^{\frac{-n}{2(b-a)^2}\epsilon^2}, && (3.12)
\end{aligned}$$

where equation (3.8) results from the definition of i^*. Equation (3.9) is a result of the probability union bound and in inequality (3.11), we used the result from Lemma 8. □

The significance of this result is that we get an exponential decay in the probability of being suboptimal as the number of sampled trees n we optimize over increases. Agreeing with these results, in practice (see section 3.9), we observe promising performance for forests even with a small number of trees. This motivates us to explore heuristics limiting the number of trees because we get diminishing returns on adding trees to the random forest when it is large, but the solve time is exponential in the number of trees due to the MILP formulation.

Next, we show that even if the number of sets M in the target random forest's

partition grows very large, the sub-optimality of the solution chosen from a sampled forest can still be bounded. More specifically, if we denote $\mathcal{M}(\epsilon) = \{i | S_{max} - S_i \geq \epsilon\}$ as the indices of sets in the partition which have a score further than ϵ away from the optimal set's score and $\mathcal{M}^C(\epsilon) = \{i | S_{max} - S_i < \epsilon\}$ as the indices of the sets in the partition which have a score within ϵ from the optimal set's score, then, so long as as $|\mathcal{M}(\epsilon)|$ grows large proportional to $|\mathcal{M}^C(\epsilon/2)|$, we can find a bound which is "constant" in the size of M. In particular, if we consider a general enough probabilistic framework behind how the scores $(S_i)_{i=1,\cdots,M}$ were generated (that for example $(S_i)_{i=1,\cdots,M}$ are iid from any particular distribution), then we show that for a fixed $\epsilon > 0$, $E[\frac{|\mathcal{M}(\epsilon)|}{|\mathcal{M}(\epsilon/2)^C|+1}] \underset{M\to\infty}{=} \mathcal{O}(1)$.[5] If we denote $\lim_{M\to\infty} E[\frac{|\mathcal{M}(\epsilon)|}{|\mathcal{M}(\epsilon/2)^C|+1}] = \alpha(\epsilon)$, then we can prove that:

$$\lim_{M\to\infty} P(S_{max} - S_{i^*} \geq \epsilon) \leq \alpha(\epsilon) e^{\frac{-n}{8(b-a)^2}\epsilon^2}.$$

To achieve this result, we need some additional assumptions on the structure of the problem including independence between the scores of sets within the same tree. While this assumption may at first glance seem strong, it is in fact easily shown that the covariance between the scores of two sets in the sampled random forest decreases linearly with the sample size. More specifically, take the example where there exist random variables X_1^t, X_2^t with covariance $COV(X_1^t, X_2^t) = \rho$, but X_1^t (and respectively X_2^t) are independent over $t = 1, ..., n$. This is analogous to having dependence between the scores of sets within the same tree, but independence between the scores of the same set over different trees (due to each tree being independently sampled from the distribution of trees). If we define $Y_1 = \sum_{t=1}^n \frac{X_1^t}{n}$ and $Y_2 = \sum_{t=1}^n \frac{X_2^t}{n}$, then $COV(Y_1, Y_2) = \rho/n$. Therefore, for large n, the correlation effect is very small, so independence is a reasonable assumption for analyzing the behavior we observe in practice.

Furthermore, it is necessary to assume that Y_i have hazard rate dominance when ordered according to their score. Since the Y_i are a function of X_i^t in our analysis, it

[5] We include the proof in appendix B.3.

is more natural to make assumptions about the distribution of X_i^t. In particular, we assume that X_i^t have increasing failure rate (IFR) and have hazard rate dominance. Lemma 3 shows that if X_i^t are IFR and there is hazard rate dominance between the X_i^t, then there is hazard rate dominance between the Y_i.

Definition 1 (Increasing Failure Rate)**.** *A continuous random variable X has an increasing failure rate iff:*

$$\frac{f_X(x)}{1-F_X(x)} \geq \frac{f_X(y)}{1-F_X(y)} \qquad \forall x \geq y,$$

where f_X and F_X are respectively the probability density function and the cumulative density function of X.

Definition 2 (Hazard rate Dominance)**.** *A continuous random variable X has hazard rate dominance over a continuous random variable Y:*

$$X \geq_{HRD} Y \Leftrightarrow \frac{f_X(x)}{1-F_X(x)} \leq \frac{f_Y(x)}{1-F_Y(x)} \qquad \forall x \in (-\infty, \infty),$$

where f_X and F_X are respectively the probability density function and the cumulative density function of X.
This is also equivalent to:

$$X \geq_{HRD} Y \Leftrightarrow P(a \geq Y | Y \geq b) \leq P(a \geq X | X \geq b) \qquad \forall a \geq b$$

Lemma 3 (HRD is closed under convolution)**.** *Let $X_1, X_2, \cdots, X_n$ be iid and let $Y_1, Y_2, \cdots, Y_n$ be iid such that $X_i \sim X$ and $Y_i \sim Y$. If X and Y are IFR and $X \leq_{HRD} Y$, then $\frac{1}{n}\sum_{i=1}^n X_i \leq_{HRD} \frac{1}{n}\sum_{i=1}^n Y_i$*

This is proved in Theorem $1.B.4$ in Shaked and Shanthikumar (2007). The following theorem uses these assumptions to give a result that is stronger than in Theorem 3's.

Theorem 4. *Let $\{X_i^t\}_{i=1,\cdots,M}^{t=1,\cdots,T}$ be continuous random variables that lie in an interval $[a, b]$, independent with respect to i and t, such that $\forall i$, $(X_i^t)_t$ are iid with $E[X_i^t] = S_i \; \forall t$*

and $X_i^t \leq_{HRD} X_{i+1}^t$. *Furthermore, assume that the distributions of* X_i^t *are IFR. Let* $Y_i = \sum_{t=1}^n \frac{X_i^t}{n}$ *and* $i^* = \{\arg\max_i Y_i | \omega_i \cap \mathcal{X} \neq \emptyset\}$, *while* $S_{max} = \{\max_i S_i | \omega_i \cap \mathcal{X} \neq \emptyset\}$. *Define* $\mathcal{M}(\epsilon) = \{i | S_{max} - S_i > \epsilon\}$, *and* $\mathcal{M}(\epsilon)^C = \{i | S_{max} - S_i < \epsilon\}$. *Then for any* $\epsilon > 0$,

$$P(S_{max} - S_{i^*} > \epsilon) \leq |\mathcal{M}(\epsilon)|(e^{-\frac{-n}{8(b-a)^2}\epsilon^2})^{|\mathcal{M}(\epsilon/2)^C)|} + \frac{|\mathcal{M}(\epsilon)|e^{\frac{-n}{8(b-a)^2}\epsilon^2}}{(|\mathcal{M}(\epsilon/2)^C|+1)(1-e^{\frac{-n}{8(b-a)^2}\epsilon^2})} \tag{3.13}$$

Proof. In what follows, denote f_{Y_j} the probability density function of Y_j for $j = 1, \cdots, M$ and let $\gamma = S_{max} - 3\epsilon/4$. We have :

$$P(S_{max} - S_{i^*} > \epsilon) = P(\bigcup_{j \in \mathcal{M}(\epsilon)} 1_{\{j=i^*\}}) \tag{3.14}$$

$$= P(\bigcup_{j \in \mathcal{M}(\epsilon)} Y_j \geq \max_{k \neq j} Y_k) \tag{3.15}$$

$$\leq \sum_{j \in \mathcal{M}(\epsilon)} P(Y_j \geq \max_{k \in \mathcal{M}(\epsilon/2)^C} Y_k) \tag{3.16}$$

$$\leq \sum_{j \in \mathcal{M}(\epsilon)} \int_{-\infty}^{\infty} f_{Y_j}(y) P(y \geq \max_{k \in \mathcal{M}(\epsilon/2)^C} Y_k | Y_j = y) dy \tag{3.17}$$

$$\leq \sum_{j \in \mathcal{M}(\epsilon)} \int_{-\infty}^{\infty} f_{Y_j}(y) P(y \geq \max_{k \in \mathcal{M}(\epsilon/2)^C} Y_k) dy \tag{3.18}$$

$$\leq \sum_{j \in \mathcal{M}(\epsilon)} \int_{-\infty}^{\infty} f_{Y_j}(y) \prod_{k \in \mathcal{M}(\epsilon/2)^C} P(y \geq Y_k) dy \tag{3.19}$$

$$\leq \sum_{j \in \mathcal{M}(\epsilon)} \left(\int_{-\infty}^{\gamma} f_{Y_j}(y) \prod_{k \in \mathcal{M}(\epsilon/2)^C} P(y \geq Y_k) dy + \int_{\gamma}^{\infty} f_{Y_j}(y) \prod_{k \in \mathcal{M}(\epsilon/2)^C} P(y \geq Y_k) dy \right), \tag{3.20}$$

where equation (3.14) results from the definition of the set $\mathcal{M}(\epsilon)$ (a solution is only further than ϵ away from optimality if the highest realized score is in set $\mathcal{M}(\epsilon)$) and equality (3.15) from the definition of i^*. Equation (3.16) is a result of the probability union bound and from the fact that $\mathcal{M}(\epsilon/2)^C \subseteq \{1, \cdots, M\}$. In inequality (3.17), we used conditional probability and inequalities (3.18) and (3.19) are a direct result of

the independence of $(Y_k)_{k=1,\cdots,M}$. Finally, (3.20) results from the additivity property of integrals.

Furthermore, we show using Hoeffding's inequality that for $y \leq \gamma$:

$$P(y \geq Y_k) \leq P(\gamma \geq Y_k) \leq e^{\frac{-n}{8(b-a)^2}\epsilon^2}, \tag{3.21}$$

which gives:

$$\int_{-\infty}^{\gamma} f_{Y_j}(y) \prod_{k \in \mathcal{M}(\epsilon/2)^C} P(y \geq Y_k) dy \leq (e^{\frac{-n}{8(b-a)^2}\epsilon^2})^{|\mathcal{M}(\epsilon/2)^C)|}. \tag{3.22}$$

For $y \geq \gamma$, we have that:

$$\begin{aligned}
P(y \geq Y_k) &= P(y \geq Y_k | Y_k \leq \gamma) P(Y_k \leq \gamma) + P(y \geq Y_k | Y_k \geq \gamma) P(Y_k \geq \gamma) && (3.23)\\
&= P(Y_k \leq \gamma) + P(y \geq Y_k | Y_k \geq \gamma)(1 - P(Y_k \leq \gamma)) && (3.24)\\
&= P(Y_k \leq \gamma)(1 - P(y \geq Y_k | Y_k \geq \gamma)) + P(y \geq Y_k | Y_k \geq \gamma) && (3.25)\\
&\leq e^{\frac{-n}{8(b-a)^2}\epsilon^2}(1 - P(y \geq Y_k | Y_k \geq \gamma)) + P(y \geq Y_k | Y_k \geq \gamma) && (3.26)\\
&= e^{\frac{-n}{8(b-a)^2}\epsilon^2} + P(y \geq Y_k | Y_k \geq \gamma)(1 - e^{\frac{-n}{8(b-a)^2}\epsilon^2}), && (3.27)
\end{aligned}$$

where (3.23) is a direct application of conditional probability, (3.24) and (3.25) result from some manipulations of probabilities and (3.26) results from (3.21).
From the assumptions of Theorem 3 and the result from Lemma 3, we get that $Y_j \leq_{HRD} Y_k$. In particular, for $y \geq \gamma$:

$$P(y \geq Y_k | Y_k \geq \gamma) \leq P(y \geq Y_j | Y_j \geq \gamma), \tag{3.28}$$

and so combining (3.27) and (3.28) give:

$$P(y \geq Y_k) \leq e^{\frac{-n}{8(b-a)^2}\epsilon^2} + P(y \geq Y_j | Y_j \geq \gamma)(1 - e^{\frac{-n}{8(b-a)^2}\epsilon^2}). \tag{3.29}$$

It results that:

$$\int_{\gamma}^{\infty} f_{Y_j}(y) \prod_{k \in \mathcal{M}(\epsilon/2)^C} P(y \geq Y_k) dy \tag{3.30}$$

$$\leq \int_{\gamma}^{\infty} f_{Y_j}(y) \left(e^{\frac{-n}{8(b-a)^2}\epsilon^2} + P(y \geq Y_j | Y_j \geq \gamma)(1 - e^{\frac{-n}{8(b-a)^2}\epsilon^2}) \right)^{|\mathcal{M}(\epsilon/2)^C)|} dy \tag{3.31}$$

$$= P(Y_j \geq \gamma) \int_{\gamma}^{\infty} f_{Y_j|Y_j \geq \gamma}(y) \left(e^{\frac{-n}{8(b-a)^2}\epsilon^2} + F_{Y_j|Y_j \geq \gamma}(y)(1 - e^{\frac{-n}{8(b-a)^2}\epsilon^2}) \right)^{|\mathcal{M}(\epsilon/2)^C)|} dy \tag{3.32}$$

$$= P(Y_j \geq \gamma) \left[\frac{\left(e^{\frac{-n}{8(b-a)^2}\epsilon^2} + F_{Y_j|Y_j \geq \gamma}(y)(1 - e^{\frac{-n}{8(b-a)^2}\epsilon^2}) \right)^{|\mathcal{M}(\epsilon/2)^C|+1}}{(|\mathcal{M}(\epsilon/2)^C|+1)(1 - e^{\frac{-n}{8(b-a)^2}\epsilon^2})} \right]_{\gamma}^{\infty} \tag{3.33}$$

$$= P(Y_j \geq \gamma) \frac{\left(1 - (e^{\frac{-n}{8(b-a)^2}\epsilon^2})^{|\mathcal{M}(\epsilon/2)^C)|+1} \right)}{(|\mathcal{M}(\epsilon/2)^C|+1)(1 - e^{\frac{-n}{8(b-a)^2}\epsilon^2})} \tag{3.34}$$

$$\leq \frac{e^{\frac{-n}{8(b-a)^2}\epsilon^2}}{(|\mathcal{M}(\epsilon/2)^C|+1)(1 - e^{\frac{-n}{8(b-a)^2}\epsilon^2})}. \tag{3.35}$$

Equality (3.32) results from conditional probability and the last inequality results from the direct application of Hoeffding's inequality for Y_j.

Finally, combining (3.22) and (3.35) we conclude that:

$$P(S_{max} - S_{i^*} > \epsilon) \leq |\mathcal{M}(\epsilon)|(e^{-\frac{-n}{8(b-a)^2}\epsilon^2})^{|\mathcal{M}(\epsilon/2)^C)|} + \frac{|\mathcal{M}(\epsilon)| e^{\frac{-n}{8(b-a)^2}\epsilon^2}}{(|\mathcal{M}(\epsilon/2)^C|+1)(1 - e^{\frac{-n}{8(b-a)^2}\epsilon^2})} \tag{3.36}$$

□

The IFR assumption is satisfied by a large class of commonly used distributions such as the normal, uniform, logistic and negative exponential distribution as well as the gamma, beta and weibull distributions for some classes of parameters. In addition, both these assumptions are satisfied in the case where the scores for each set in the partition $\mathcal{P}$ have an IFR distribution that is translated by some amount (i.e $f_{X_i}(x) = f_{X_{i+1}}(x + \delta_i)$). Furthermore, numerical evidence from real data (the jury

dataset) suggests that the hazard rate dominance assumption is reasonable for Y_i for large n (see appendix B.4). This intuitively makes sense because Y_i converges almost surely to S_i according to the strong law of large numbers.

3.6.1 Cross-validation for optimization

The knowledge that the sub-optimality of a random forest decays exponentially with the number of trees motivates an algorithm which uses fewer trees. This is because we get diminishing returns on adding trees to the random forest when it is large, but the solution time is exponential in the the number of trees due to the MILP formulation (see Section 3.9). Based on this logic we propose a heuristic where we decompose a large target random forest into several smaller random forests. For each random forest subset, we select the optimal solution using our MILP formulation. We then test each solution against the forests it was not optimized over, to get the average score referred to from now on as the *out-of-forest score*. We then pick the solution which has the best out-of-forest score. The rationale behind this heuristic is that we can solve many smaller MILPs much faster than we can solve one big MILP. Furthermore, we can pick a solution which is robust relative to different forests, mitigating the effects of overfitting due to using a limited number of trees in the optimization. We describe the procedure below more precisely:

Algorithm 3 cross-validation for Optimization

1: *Train* τ *forests* $R_1(\cdot), ..., R_\tau(\cdot)$ *on the data*

2: *Find optimal solution for each forest* $x_i = \arg\max_{x \in \mathcal{X}} R_i(x) \qquad \forall i \in \{1, .., \tau\}$

3: *Calculate* $\bar{R}(x_i) = \frac{1}{\tau} \sum_j R_j(x_i) \qquad \forall i \in \{1, .., \tau\}$

4: *Pick* $x^{CV} = \arg\max_{i \in \{1,..,\tau\}} \bar{R}(x_i)$

We can derive analytical bounds on the performance of this algorithm. Due to a random forest being the average of several trees (or equivalently average of smaller random forest), the global optimal solution is $z^* = \max_{x \in \mathcal{X}} \frac{1}{\tau} \sum R_i(x)$.

Proposition 2. *Let x^{CV} be the solution derived from the cross-validation for optimization procedure, over τ forests $R_1, ..., R_\tau$. Then*

$$\frac{1}{\tau}\sum_{i=1}^{\tau} R_i(x^{CV}) \leq z^* \leq \frac{1}{\tau}\sum_{i=1}^{\tau} \max_{x\in\mathcal{X}} R_i(x)$$

We leave the proof for appendix B.5. These bounds are very useful in practice; if we solve the heuristic for forests of a certain size, we know how good the obtained solution is relative to the best solution, and how much we could potentially gain by repeating the heuristic with larger forests. This allows the practitioner to make an informed decision about whether to increase the forest size and the computational burden.

Using the same assumptions about tree generation as were previously used, we can prove that the sub-optimality of the solution obtained from cross-validation decays exponentially with both the number of trees n in each subset and the number of subsets used for cross-validation τ. Using the same notation, let $i_1^*...i_\tau^*$ be the index of the partition obtained by optimizing over a random forest approximation. Let i^{CV} be the index of the partition chosen by using cross-validation over subsets.

Corollary 2. *For all subsets $\{j = 1, ..., \tau\}$, trees $\{t = 1, ..., n\}$, and partitions of the target forest $\{i = 1, .., M\}$, let $\{X_{ij}^t\}_{i=1,\cdots,M;j=1,\cdots,\tau}^{t=1,\cdots,T}$ be independent random variable bounded by the interval $[a, b]$, with $E[X_{ij}^t] = S_i$. Let $Y_{ij} = \sum_{t=1}^{n} \frac{X_{ij}^t}{n}$ and $i_j^* = \{\arg\max_i Y_{ij} | \omega_i \cap \mathcal{X} \neq \emptyset\}$, while $S_{max} = \{\max_i S_i | \omega_i \cap \mathcal{X} \neq \emptyset\}$, and $i^{CV} = \arg\min S_{max} - S_{i_j^*}$. Define $\mathcal{M}(\epsilon) = \{i | S_{max} - S_i > \epsilon\}$. Then for any $\epsilon > 0$, $P(S_{max} - S_{i^{CV}} > \epsilon) \leq (|\mathcal{M}|(\epsilon)e^{\frac{-n}{2(b-a)^2}\epsilon^2})^\tau$*

We leave the proof for appendix B.5.

We now proceed by introducing two case studies with real data which both analyze the behavior of the algorithms we have introduced, and explore the effectiveness of random forest compared to optimizing over different objective functions.

3.7 Case study: property investment

Property investors face difficulty in assessing how much a property will sell for on the market. Residential properties have many different features (such as the number of bedrooms, bathrooms, floors, size of house, quality of construction, location) which are prioritized and valued in different ways by different buyers. Furthermore, there is a degree of unpredictability in the sale price depending on which buyers happen to be interested in a property. This makes it hard for property investors to make sound investment decisions when evaluating different options they might want to take. In particular, we look at a situation in which the property investor is interested in buying an empty lot, constructing a new house then selling that property to make a profit. We help overcome this uncertainty by estimating the sale price for a given property using a random forest model trained from data on previous house sales. We then optimize to find the property which is predicted to achieve the highest price, as predicted by the random forest.

3.7.1 Data

We have data from Kaggle on house sale prices for King County, which includes Seattle (Harlfoxem (2016)). It includes houses sold between May 2014 and May 2015. The dataset contains 19 features, and 21613 observations of houses sold. We divide the features into those related to the house, and those related to the lot. We removed the features which a property investor would not be able to control (i.e was the house viewed before sale), leaving us with the variables in table 3.2.

House Variables	Meaning	**Lot Variables**	Meaning
bedrooms	number of bedrooms	**zipcode**	zipcode house is located in
bathrooms	bumber of bathrooms	**lat**	latitude of house
size	square footage of the home	**lon**	longitude of house
base	square footage of the basement	**sqftlot**	square footage of the lot
grade	quality of construction	**water**	view of waterfront (binary)

Table 3.2: Variables of housing dataset

3.7.2 Prediction

Table 3.5 shows the performance of various algorithms for predicting the sale price of a property. We used a 70% / 30% training / testing split. We see that XGBoost (another ensemble tree approach) performs the best with 90% OOS accuracy while Random Forest performs second best with 88% OOS accuracy. Regression approaches do worse, achieving around 70% OOS accuracy.

Model	Training accuracy (R^2)	Testing accuracy (R^2)
Random forest	98	88
XGBoost	100	90
CART	99	76
Ridge Regression	70	70
Linear Regression (OLS)	70	69

Table 3.3: Predictive accuracy house sales prices

3.7.3 Formulation

Suppose we have a number of empty sections the investor has to choose between. The investor needs to decide i) which lot to purchase and ii) what kind of house to build on that lot. We assume the property investor has a fixed budget with which to buy the lot and construct the house. The property investor wants to maximize the sale price (or predicted value) of the resulting house.

Suppose we have trained a random forest $R(\mathbf{x^H}, \mathbf{x^L})$ from data to predict the property sale price, where $\mathbf{x^H} = (\mathbf{x^H_{bed}}, \mathbf{x^H_{bath}}, \mathbf{x^H_{size}}, \mathbf{x^H_{grade}}, \mathbf{x^H_{base}})$ are the features of the house, while $\mathbf{x^L} = (\mathbf{x^L_{zip}}, \mathbf{x^L_{lat}}, \mathbf{x^L_{lon}}, \mathbf{x^L_{sqftlot}}, \mathbf{x^L_{water}})$ are the features of the lot, corresponding to table 3.2. We want to maximize the sale price by maximizing $R(\mathbf{x^H}, \mathbf{x^L})$ as our objective. We assume that $R(\cdot, \cdot)$ has been trained a priori on a housing dataset and is treated as an input for the optimization model. We have $x^H_{bed}, x^H_{bath} \in \mathbb{Z}^+$, $x^H_{size}, x^H_{base}, \in \mathbb{R}^+$ and a binary vector $x^H_{grade} \in \{0,1\}^8$ denoting which grade of materials the house is built out of.

Suppose there is a set $\mathcal{A}$ of available lots the property investor has to choose from. Each lot l has a vector of features $\boldsymbol{\theta}$ which describes the lot (size, zip code etc - see

table 3.2). Let L be the number of features of the lot. Define z_{lot} as a binary variable, with $z_{lot} = 1$ if we choose lot l, 0 otherwise. Constraint (3.38) assigns the features of the chosen lot to the vector $\mathbf{x^L}$. Constraint (3.39) specifies that only one lot is chosen.

Let c_{lot} be the cost of purchasing a lot, c_{bath} and c_{bed} be the cost of building each bathroom and bedroom respectively. Let c_{base} be the cost per square foot of constructing a basement, c_{grade} be the cost per square foot of constructing a house with a given quality grade. Constraint (3.40) requires that the cost of purchasing the lot and constructing the house is less than a given budget B. We have an additional constraint (3.41) requiring that the size of the house is smaller than the size of the lot. Note, in our actual implementation we linearized $x_{grade}x_{size}$ by introducing an additional variable $x_{grade,size} = x_{grade}x_{size}$ using standard MIP techniques.

We also require the house chosen to be within the convex hull of the set of properties observed in the data. This ensures the feasible set is bounded, and that the selected house is not too dissimilar from the houses observed in the dataset. Let $\{\tilde{x}^i\}_{i=1}^m$ be the features of the m houses in the dataset the random forest was trained on. This restriction is enforced in constraint (3.43) and (3.44).

$$\underset{x^H, x^L, z}{\text{maximize}} \quad R(\mathbf{x^H}, \mathbf{x^L}) \tag{3.37}$$

$$\text{subject to} \quad x_i^L = \sum_{lot \in \mathcal{A}} \theta_{lot,i} z_{lot} \quad \forall i \in \{1, .., L\} \tag{3.38}$$

$$\sum_{lot \in \mathcal{A}} z_{lot} = 1 \tag{3.39}$$

$$c_{bed}x_{bed}^H + c_{bath}x_{bath}^H + c_{base}x_{base}^H + \sum_{lot \in \mathcal{A}} c_{lot}z_{lot} + \sum_{grade=1}^{12} c_{grade}x_{grade}^H x_{size}^H \leq B \tag{3.40}$$

$$x_{size}^H \leq x_{sqftlot}^L \tag{3.41}$$

$$\sum_{grade=1}^{12} x_{grade}^H = 1 \tag{3.42}$$

$$x_j = \sum_{i=1}^{m} \lambda_i \tilde{x}_j^i \qquad \forall j \in \{bed, bath, size, floor, grade, base\} \tag{3.43}$$

$$\sum_{i=1}^{m} \lambda_i = 1 \tag{3.44}$$

$$x^H_{grade}, z_{lot} \in \{0,1\}, x^H_{bed}, x^H_{bath} \in \mathbb{Z}^+, x^H_{size}, x^H_{base} \in \mathbb{R}^+ \tag{3.45}$$

This problem requires cost parameters which weren't given as part of the data. We have made efforts to find these costs where available through discussions with construction professionals and research, but acknowledge additional estimation would be required to use this model in practice. Furthermore, there are additional complexities involved with pricing and building a house (for example, the square meter costs observed in practice are likely to be slightly concave due to economies of scale), but we believe this formulation provides a simple but illustrative example of the type of problems the random forest optimization is able to solve. Also the primary goal of this case study is to compare the performance of the different algorithms we discuss in this chapter. In our experiments, the comparative insights held over a wide range of parameters, although individual solutions of course varied. This model could be customized to reflect different scenarios faced by the property investor, with additional constraints and more accurate predictions of parameter costs introduced. In our experience, mixed-integer programming provides a lot of flexibility in being able to model such constraints.

We take the cost per square meter for each grade as $4 = \$88.45, 5 = \$98.30, 6 = \$109.03, 7 = \$126.59, 8 = \$148.11, 9 = \$169.73, 10 = \$231.25, 11 = \$330.81, 12 = \$380.45$ and the cost of the basement $c_{base} = \$60$ per square foot. It is not legal to build a new house with grade less than 4. These costs are taken from the Craftsman National Building Cost Manual Moselle (2015) associated with building a 6 corner 2000 square foot house, with the descriptions of the grades matched to those described by the King County local government. We use cost per bathroom $c_{bath} = \$20,000$, cost per bedroom $c_{bed} = \$10,000$. We use a constant cost per lot $c_{lot} = \$250,000$. To obtain the set of available vacant lots, we randomly sample properties from the dataset of house sales, and extract the features relevant to the lot. We assume the budget available is $B = \$450,000$.

3.8 Jury selection case study

3.8.1 Background

Juries are composed of members of the public and decide many court cases. While these jurors are supposed to be impartial and make their decision based solely on the available evidence, several studies have suggested that jurors may have inherent biases based on their race, education, wealth and political views (Denove and Imwinkelried (1995); Moran and Comfort (1982); Bornstein and Rajki (1994)). In many legal systems (including in the US), lawyers are allowed a fixed number of peremptory challenges, whereby they are allowed to reject potential jurors without stating a reason. Lawyers are also able to ask jury members a number of questions to ascertain personal information about them. This has spawned an industry known as jury consulting (a branch of trial consulting), where experts use a range of qualitative social science and psychology techniques to try and choose favorable juries for a civil or criminal trial. Despite a lack of empirical evidence supporting its efficacy, the trial consulting industry was estimated to be worth 400 million in 1999, with over 400 firms operating in the market (Hartje (2004)).

We study the problem of selecting a jury from a larger juror pool, taking into account information about the demographics and views of the eligible jurors and characteristics of the case. First we will show that it is possible to predict the verdict of a jury with a higher accuracy than 50%, solely based on this information, then we will show how to combine these predictive models with optimization to pick these jurors in an intelligent way.

3.8.2 Predicting jury verdict

In this section, we explore how the members of a jury influence the verdict. Specifically, we use real data on jurors' personal demographics and views, and the characteristics of the case to predict whether the jury will return a guilty or not guilty verdict.

Data

We obtained data from the Inter-university Consortium for Political and Social Research on non-capital felony jury trials in the state trial courts of Los Angeles (CA), Maricopa (AZ), Bronx (NY), and Washington D.C. (Hannaford-Agor et al. (2003).) The information was collected for 3,497 jurors and 354 cases with 83 features. Some of the features are related to jurors' opinions on the trial and their interpretation of evidence. Since this information is clearly not available before the trial starts, we restricted the dataset to 20 variables that the interviewing lawyer could easily collect prior to the trial. Table 3.4 shows a list of these variables.

Jury Variables	Meaning	**Case Variables**	Meaning
police	Trust the police	**deftgend**	Defendant's gender
amcourts	Trust the courts	**defrace**	Defendant's race
crmsevre	Think crime serious problem	**victgend**	Victim's gender
previous	Been a juror before	**victrace**	Victim's race
civilcrm	Previous jury-criminal of civil	**victrela**	Victim's relationship to defendant
gender	Gender	**casetype**	Case type
age	Age		
school	Highest level of education		
race	Race/ethnicity		
religion	Strength of religious beliefs		
income	Household income		
jobstat	Job status		
occupat	Occupation		

Table 3.4: Variables of our dataset

Predictive accuracy

Model	Training accuracy (%)	Testing accuracy (%)
Random forest	59	60
XGBoost	100	57
CART	85	51
SVM	53	53
Logistic Regression	65	56
Linear Regression	53	49
Random forest with only case features (baseline)	55	54

Table 3.5: Predictive jury verdict averaged over 20 data splits

Table 3.5 shows the performance of various algorithms for predicting jury outcomes. We used a 70% / 30% training / testing split and averaged our trials over 20 random splits of the data. We see that random forest performs the best with 60% OOS accuracy while XGBoost performs second best with 57% OOS accuracy. To es-

timate the increase in accuracy attributable to having information about the jury and not due to other features of the case, we compared against a baseline random forest which had the jury features omitted. This is a strong baseline to compare against, but we still managed to get a 6% improvement OOS. This is surprising considering we do not use any evidence based features in our predictions.

3.8.3 Jury selection formulation

In the jury selection problem, the lawyer has to select a subset $\mathcal{S}$ of jurors from a larger pool of available jurors $\mathcal{A}$. In most jurisdictions the number of jurors chosen is 12 ($|S|= 12$). In all generality, the probability a jury will return a guilty verdict is a function of the set of jurors chosen, and the inherent biases they might poses. Suppose each juror j has a vector of features $\boldsymbol{\theta_j}$ (education, religion etc - see table 3.4) which are relevant in predicting their likelihood of voting guilty. We assume the probability of a jury returning a guilty verdict can be modeled using a function $R(\boldsymbol{x^J}, \boldsymbol{x^C})$ where $\boldsymbol{x^J}$ are jury features, which are additive in the features of the jurors it comprises of $\boldsymbol{x^J} = \sum_{j \in S} \boldsymbol{\theta^j}$. $\boldsymbol{x^C}$ is case dependent data which the lawyer has no control over, but still may affect the outcome of the case. An example of this is the gender of the victim, which the data suggests interacts with the gender of the juror to influence the verdict. We will restrict our attention to when $R(\boldsymbol{x^J}, \boldsymbol{x^C})$ is a random forest.

There are two main objectives that could be considered when choosing a jury: i) choosing the most fair jury or ii) choosing a jury which is most likely to return an outcome desired by the lawyer. Both objectives can be modeled under our framework. To align with the lawyer's goals, we would either maximize the probability of returning a guilty verdict by setting the objective to be $\max R(\boldsymbol{x})$ as a prosecuting lawyer, or minimize the probability of returning the guilty score by setting the objective to be $\min R(\boldsymbol{x})$ as a defending lawyer. To align with the state's goal of selecting the most fair jury, we could set the objective to be $\min|R(\boldsymbol{x}) - \bar{R}|$, where $\bar{R}$ is the probability of a jury selected from the population at random returning a guilty verdict. We define z_j as a binary variable, with $z_j = 1$ if we choose to include juror j in the jury, 0

otherwise, and let f be the number of features of the jury we have data on. We can formulate the jury selection problem for the prosecuting lawyer as follows:

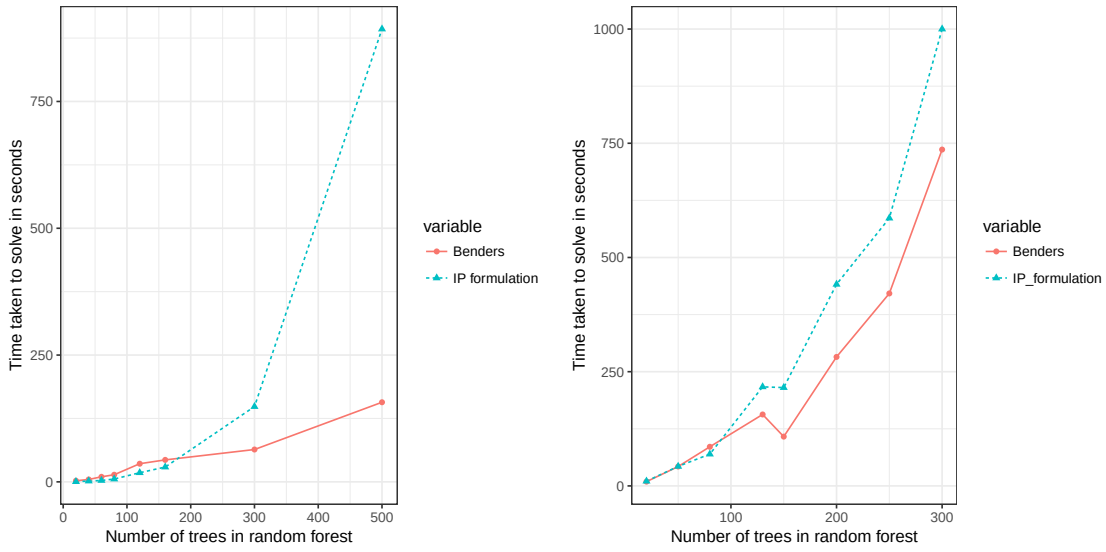
$$\begin{aligned} \underset{x^J,z}{\text{maximize}} \quad & R(\boldsymbol{x}^J, \boldsymbol{x}^C) \\ \text{subject to} \quad & x_i^J = \sum_{j \in \mathcal{A}} \theta_{ij} z_j \quad \forall i \in \{1, .., f\} \\ & \sum_{j \in \mathcal{A}} z_j = 12 \\ & z \in \{0,1\}^{|\mathcal{A}|} \end{aligned} \tag{3.46}$$

3.9 Numerical experiments

In this section, we explore the performance of the various random forest optimization procedures we have proposed in sections 3.5 and 3.6.1 on jury and housing data. Note that this section is a comparison of how well the algorithms do in optimizing over a given random forest, rather than an analysis of how good random forest optimization is at selecting solutions, which we explore in section 3.10. We first test the speed of Benders decomposition compared to the standard MIP formulation, then test the accuracy and speed of the cross-validation performance compared to the MIP. We test the sensitivity of the algorithms discussed with the key parameter of our model; the number of trees in the random forest.

3.9.1 Testing environment

The random forests are trained on the datasets in `Python` using `Scikit-learn` `RandomForestClassifier` and `RandomForestRegression` from (Pedregosa et al. (2011a)) for jury and property selection respectively. The optimization formulations were coded in `Python` and solved using a `Gurobi 6.5` solver. For the jury selection problem, the juror pool size used was 20, while in the property investment problem the number of vacant lots to choose between was 10.



(a) On the jury dataset

(b) On the housing dataset

Figure 3-3: Time taken to solve algorithms for increasing forest size

3.9.2 Solution time

The solution times of the different algorithms are shown in Figure 3-3. As expected, the MIP formulation displays exponential behavior with respect to the number of trees in the random forest. The algorithm is very fast for small to medium sized problem instances up to 160 trees on the jury dataset (9100 nodes) and 80 trees on the housing dataset (35840 nodes) but takes prohibitively long for forests larger than this. For random forest instances larger than 500 trees on the jury dataset and 300 trees on the housing dataset, the MIP did not solve within the the imposed solve limit of 1000 seconds. The MIP formulation with Benders cuts is actually slower than the MIP formulation for a small number of trees, for forests larger than 200 trees on the jury dataset and 100 trees on the housing dataset, Benders is faster. The initial lag of the Benders formulation might be due to Gurobi's effective presolve techniques which are unable to be applied to the MIP formulation with Benders cuts because the master problem starts with very few constraints. As the problem grows larger, the benefits of Benders cuts outweigh the presolve. Interestingly, over the range of forest sizes experimented on the jury dataset, Benders looks relatively linear in the solution time, although this doesn't hold for the housing dataset.

Table 3.6: Cross-validation performance on the jury dataset

# trees target	80 ($N_{nodes} = 4537$)				300 ($N_{nodes} = 16238$)			
# subsets	20	10	5	2	20	10	5	2
# trees per subset	4	4	4	4	15	15	15	15
% optimality	96.25%	96.25%	91.25%	66.25%	98.67%	98.67%	96.67%	81.67%
% time taken	5.55%	3.12%	1.74%	0.92%	12.51%	7.18%	5.12%	3.32%

Table 3.7: Cross-validation performance on the housing dataset

# trees target	40 ($N_{nodes} = 66848$)				60 ($N_{nodes} = 96792$)			
# subsets	10	5	2	1	10	5	2	1
# trees per subset	4	4	4	4	6	6	6	6
% optimality	98.5%	96.9%	83.6%	92.8%	100.0%	99.1%	93.9%	93.9%
% time taken	89.0%	45.1%	17.4%	8.7%	74.0%	35.4%	13.6%	6.7%

3.9.3 Cross-validation experiments

For each of these experiments, there is a target forest to optimize over, with the number of nodes in that forest given by N_{nodes} (this is an indication of the size of each tree). We compare solving the MIP for this target forest, versus splitting the forest into a number of subsets, solving each one using the MIP, and using cross-validation to find the best solution.

Tables 3.6 and 3.7 show the performance of the cross-validation algorithms relative to the MIP formulation, both in terms of the percentage of time taken to solve, and score of the cross-validation solution as a percentage of the optimal MIP solution for the jury and housing datasets. For completeness, we have further results for a greater range of trees in appendix B.7. We observe that the cross-validation procedure is able to achieve a large percentage of optimality in a fraction of the time taken to solve the full MIP. This is surprising, since the forests that cross-validation is optimizing over are just one tenth or twentieth of the size of the target forest for the jury and housing datasets respectively. Cross-validation is particularly effective for the jury dataset, where the solve times are less than 10% of the MIP, but achieve over 95% of the accuracy if parameters are chosen correctly. This is because there is high variance in the predictions associated with the jury dataset (each forest subset has comparatively low accuracy), so there is significant value in validating the score of the jury chosen

using different forests to the one it was chosen with. This suggests more broadly that cross-validation is more valuable on noisy datasets where it is difficult to train accurate models. We further explore the behavior of cross-validation with additional numerical experiments in appendix B.7.

3.10 Comparison with optimizing over other machine learning objective functions

3.10.1 Testing environment

Clearly we cannot observe the actual outcome for the solutions we select with our optimization procedure, so we need a way to estimate the effectiveness of the random forest optimization methodology compared to other choices of machine learning objective functions. In predictive machine learning tasks, in-sample tests are not sufficient evidence of performance because one may overfit the data and inflate the performance metrics, so it is good practice to do an out-of-sample test on a different dataset. Likewise, for our problem, we propose a similar process to assess the performance of the random forest optimization problem. For comparison, we test against solutions obtained through optimizing over other machine learning objective functions using algorithm 4.

Algorithm 4 Out-of-sample testing procedure

1: Split the jury dataset into a training set and testing set.
2: Train a set of selection machine learning functions $F_i^{train}(x),\ i = 1, ..., m_{train}$ from the training data.
3: Train a set of evaluation machine learning functions $F_j^{test}(x),\ j = 1, ..., m_{test}$ from the test data.
4: **for** k in $\{1, ..., K\}$ **do**
5: Select an optimal solution $x_i^* = \arg\max_{x \in \mathcal{X}} F_i^{train}(x)\ \forall i = 1, ..., m_{train}$
6: Evaluate solution OOS with different evaluators $F_j^{test}(x_i^*),\ \forall i = 1, ..., m_{train},\ \forall j = 1, ..., m_{train}$
7: Take average over K trials.

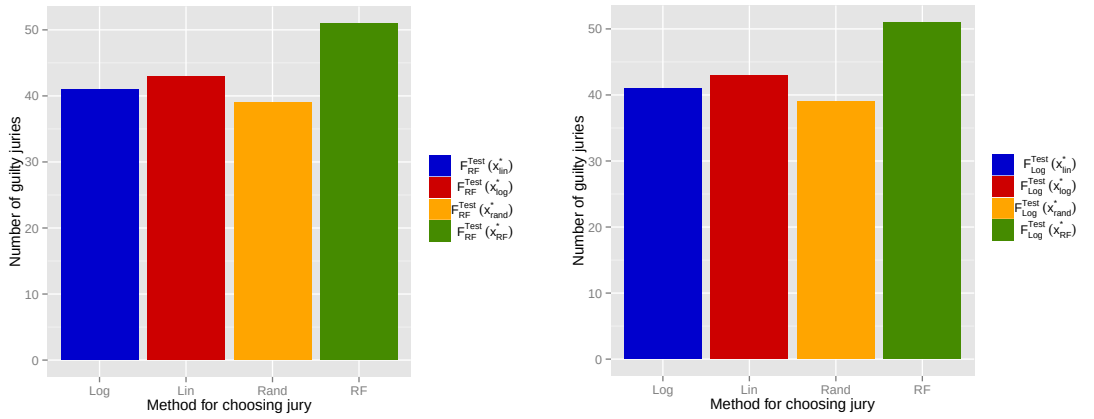
We train multiple machine learning functions from the training data $F_i^{train}(x),\ i =$

$1...m_{train}$ and choose the best solution as evaluated by each random forrest function $x_i^* = \arg\max_{x \in \mathcal{X}} F_i^{train}(x)$. This is to compare the random forest solution with the solution that would be found if we optimized over a different machine learning function. Clearly there are many considerations that need to be taken when choosing a machine learning function to optimize over, including the predictive accuracy of the machine learning function and the difficulty of the resultant optimization problem.

For the jury selection (classification) problem, we compare the juries selected using a random forest, a linear regression and a logistic regression objective function respectively $F_{RF}^{train}(x), F_{Lin}^{train}(x), F_{Log}^{train}(x)$ to obtain $x_{RF}^*, x_{Lin}^*, x_{Log}^*$. For more details on optimizing over a logistic regression objective, we refer the reader to appendix B.6. For the property investment (regression) problem, we select vacant lots and build houses using a random forest and linear regression objective function $F_{RF}^{train}(x), F_{Lin}^{train}(x)$ to obtain x_{RF}^*, x_{Lin}^*.

Next we need to measure the performance of these solutions. We propose evaluating the performance of each solution using a range of different machine learning functions trained on an independent testing dataset. Using a different testing dataset allows us to check for overfitting on the training set. Using different functions allows us to check for model misspecification (i.e what if we use a linear model and the data has non-linear relationships?), and also allows us to use high accuracy predictors which we might not be able to optimize over.

For the jury selection problem, we evaluate the juries chosen using a random forest evaluator $F_{RF}^{test}(x)$ and a logistic regression evaluator $F_{Log}^{test}(x)$. We choose the random forest as a test classifier because it has the highest predictive accuracy and choose the logistic regression test classifier to avoid any possible bias in using a random forest structure for training and testing. For the property investment problem, we evaluate the houses using a random forest and XGBoost objective function $F_{RF}^{test}(x), F_{XGB}^{test}(x)$ due to the predictive accuracy of these models. To give us a reasonable sample size, we set $K = 100$ (repeat this procedure 100 times for different juror pools and different vacant lots).



(a) Random forest evaluator $F^{test}_{RF}(\cdot)$ (b) Logistic regression evaluator $F^{test}_{Log}(\cdot)$

Figure 3-4: Number of juries (out of 100) which are predicted to return a guilty verdict according to out-of-sample model, if optimized for guilty verdict

3.10.2 Results

Jury selection

Figure 3-4 shows the performance of juries chosen via various approaches[6]. For this simulation, we assume the optimizer is a prosecution lawyer who is trying to get a guilty verdict. Figure 3-4 shows the number of juries which were classified as guilty by the out-of-sample evaluators out of the 100 randomly drawn juror pools. We observe that optimization over the random forest objective outperforms optimization over the other objectives when evaluated using both the random forest and logistic regression evaluators. Of particular interest is that the random forest optimization is able to outperform the logistic regression even using the logistic regression evaluator. We have a number of hypotheses as to why this is true. First, the random forest has a higher out-of-sample predictive accuracy (60%) than either logistic regression (56%) or linear regression (49%), so it is natural to assume that it can pick better juries given increased predictive power. Interestingly, the improvement in the number of convictions using random forest (+ 10%) is greater than the improvement in predictive

[6]We fixed the case features for each of the trials we selected a jury for, and selected a juror pool by bootstrapping from the juror population. Note that the results are for a difficult instance of the problem; the case type makes a conviction unlikely, so the appropriate benchmark is the likelihood of conviction for a randomly selected jury, not 50%.

Table 3.8: Difference in sale price of the property chosen using random forest (RF) vs linear regression as evaluated on OOS XGBoost

Number of lots to choose between	10 lots	20 lots	40 lots
Average improvement using RF	9.08 %	16.9 %	20.1 %
p value for significance of difference	1.5×10^{-6}	8.0×10^{-13}	6.4×10^{-15}

Table 3.9: Difference in sale price of property chosen using random forest (RF) vs linear regression as evaluated on OOS random forest

Number of lots to choose between	10 lots	20 lots	40 lots
Average improvement using RF	11.4 %	19.2 %	25.6 %
p value for significance of difference	1.5×10^{-7}	3.1×10^{-14}	1.1×10^{-15}

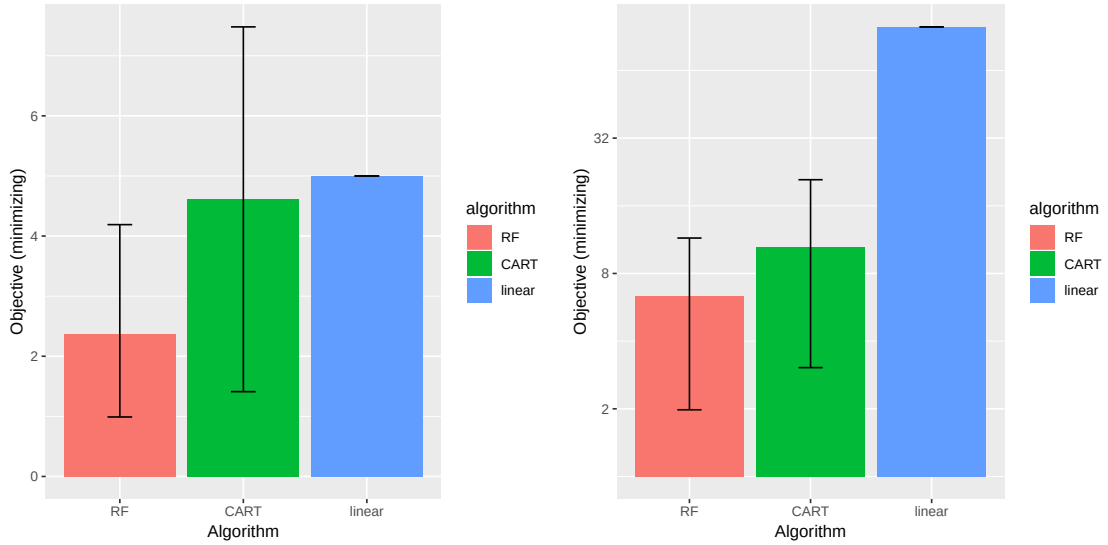
accuracy using a random forest (+4%). We hypothesize that there is a secondary benefit of using random forests, which is that there is less overfitting (of the solution to an uncertain objective) on an ensemble objective function.

Property investment

Table 3.8 shows the sale price of property chosen using random forest minus the sales price of a property chosen by linear regression, as evaluated on an out-of-sample XGBoost $F^{test}_{XGB}(x^*_{RF}) - F^{test}_{XGB}(x^*_{lin})$. Likewise Table 3.9 evaluates using an out-of-sample random forest $F^{test}_{RF}(x^*_{RF}) - F^{test}_{RF}(x^*_{lin})$.

The results shown are averaged over 100 different sets of vacant lots being available, and repeated for the property investor being able to choose from a set of 5, 10 or 20 vacant lots. We observe improvements when choosing properties using a random forest objective compared to a linear regression objective for each lot choice size and for both evaluators. The performance appears slightly better when evaluated on random forest compared to XGBoost. We observe the difference increases with the number of lots we consider, from around 3% with 5 lots to over 15% with 20 lots. This is because the random forest is particularly good at selecting the valuable properties due to the non-linear relationship between latitude, longitude and price.

We believe these experiments show the clear advantages to optimizing using a random forest objective function compared to other machine learning functions.



(a) Friedman 1 dataset

(b) Friedman 2 dataset

Figure 3-5: Performance on synthetic data

3.10.3 Synthetic data

To overcome the issue of not being able to observe the counterfactuals in real datasets, we also ran experiments on synthetic data. With these datasets, it is possible to evaluate exactly how well each option does since we know what the counterfactuals are. We used the synthetic datasets from Friedman et al. (1991), which is a well known benchmark dataset in the machine learning community. The data sets tested are:

- Friedman 1: $y(X) = 10\sin(\pi X_0 * X_1) + 20(X_2 - 0.5)^2 + 10X_3 + 5 * X_4 + N(0,1)$ with $X_i \sim U(0,1), ß \in \{1, ...10\}$, where $X_5, .., X_{10}$ are additional noise.

- Friedman 2: $y(X) = (X_0^2 + (X_1 * X_2 - 1/(X_1 * X_3))^2)^0.5 + N(0,1)$, with $X_0 \sim U(0,100),\ X_1 \sim U(40\pi, 560\pi),\ X_2 \sim U(0,1),\ X_3 \sim U(1,11)$

We compare between learning and optimizing over a linear function, a CART tree and a random forest with 10 trees. 2000 data points are used for training the models. In both cases, we solve $\min_x y(x)$, subject to the constraint that x must be in the domain of the data (i.e $0 \leq X_0 \leq 100$ for the Friedman 2 dataset). We then find the

optimal solution x^i_{opt} for sample, and calculate the true outcome $y(x^i_{opt})$. We repeat this process 100 times to get a distribution over $y(x^i_{opt})$, which is shown in figure 3-5. The confidence bars are at 10% and 90% respectively. As can be seen in figure 3-5, the random forest typically performs better than the CART, which in turn is better than the linear regression. There also appears to be less variance in the solution of the random forest relative to the CART model. The linear regression has no variance, achieving the same solution every time, although that solution is often poor. An issue with using a linear objective function is that the solution will always be at the edge of the feasible set, even when the optimal solution is at the interior.

3.11 Conclusions

In this chapter, we examine a data-driven optimization approach, where an uncertain objective is estimated according to a machine learning function. In particular, we have shown that it is possible to optimize over a random forest objective function with general polyhedral constraints using Mixed Integer Linear Programming. While the efficacy of this approach depends on the predictive accuracy of random forest relative to other predictive models, and solution time, this approach has many advantages. First, random forests are a very powerful prediction algorithm which can fit unknown non-linear and complex interactions of features with minimal feature engineering. Second, Random forests often perform very well in practice, with high prediction accuracy. Third, we can leverage existing MILP solvers which are effective at solving large scale complex problems and provide flexibility to model a wide range of problems. We demonstrate the effectiveness of this approach at solving real world problems using a case study on jury selection and property investment, and show this method outperforms other machine learning objective functions. We also show that it is possible to achieve significant speed improvements using Benders cuts for large scale problems.

We present analytical results showing that if we approximate a large random forest with a limited number of trees, and optimize over that sub-forest, the relative sub-

optimality of the solution decreases exponentially as a function of the number of trees. These insights lead us to develop a heuristic approach which uses a limited number of trees. The heuristic is inspired by cross-validation but we apply it to an optimization setting. We can obtain bounds on the heuristic and show that the improvement increases exponentially with the number of subsets used in cross-validation. We show that these heuristics perform well on the jury and housing data, getting solutions close to the MILP in a fraction of time.

Chapter 4

Dynamic routing with tree based value function approximations

4.1 Introduction

Many real-world, online decision making problems are too complex to solve tractably using traditional dynamic programming methods. This has led to the development of approximate dynamic programming and reinforcement learning approaches, where the future reward resulting from a decision, known as the cost-to-go function, is approximated rather than calculated exactly. Recently, machine learning techniques have been successfully used to approximate this function using data generated by following a policy. In particular, the use of neural networks (deep reinforcement learning), has led to success and break-throughs in a variety of domains including beating the world's best humans in the game Go (Silver et al. (2016)) and Atari video games (Mnih et al. (2015)). Machine learning functions are successful in this setting because they are able to capture the complex non-linear interactions between the state, action and corresponding outcome with a high degree of accuracy. In these instances and many other problems, the action space (the number of possible actions that can be made at each stage) is relatively small, so enumerating all possible actions from a state is not an issue.

In comparison, in many real world operational problems, complex decisions need

to be made at every stage of an online process. These decisions are often high dimensional or combinatorial in nature and subject to constraints on what is a feasible decision. This results in a very large action space. As a result, enumeration is no longer a tractable option. The resulting problem, being combinatorial in nature, lends itself to combinatorial optimization techniques to solve it. An example which we will focus on in this chapter is the nurse practitioner routing problem. This problem is a variant of a dynamic vehicle routing problem where at every stage a routing decision needs to be made about which nurse practitioners will serve which patients. The different permutations of patients to nurse practitioner allocations is unfortunately, beyond enumeration. This is compounded by the fact that the algorithms used in dynamic programming (such as value and policy iteration) and reinforcement learning are iterative in nature and require that these optimization sub-problems are solved a large number of times over a very large state space. As a result, it is very important to be able to solve these subproblems fast.

While there are sophisticated optimization techniques for solving these subproblems on their own efficiently, difficulty arises when a machine learning function is introduced into the objective of each subproblem to describe the future reward resulting from a decision. In particular, the future reward is a *function* of the decision. Because the machine learning function is often complex and non-linear this means the subproblem is considerably more difficult to solve and the techniques typically used to solve the subproblem may no longer apply. This is particularly pronounced when the function is a highly non-linear neural network. Although there has been recent work by (Anderson et al. (2018)) on optimizing decisions which are a function of a neural network, this is well known to be very difficult task. In this work, we explore using trees and tree ensembles to approximate the cost to go function. We show how to solve the resulting subproblems efficiently using Mixed Integer Optimization (MIO) techniques, while incorporating the combinatorial constraints of the nurse practitioner routing problem. This approach has a number of advantages. The trees are able to capture the inherently non-linear behavior in the nurse practitioner routing problem, where the location of nurse practitioners, how close they are to each other and how

well they match demand, determines the effectiveness of the system over future periods. This results in significant improvements over widely used greedy approaches and linear approximations of the cost-to-go function. Furthermore, regularization and pruning of trees results in selection of a value function of appropriate complexity in the sense that the trees only reach a depth in which there is a significant difference in outcome between states. Indeed in all generality, a tree can model a look-up table for the entire state space, but often an approximation will suffice. Finally, trees are beneficial for their interpretability, so the resulting cost-to-go functions can be "interrogated" to see how the state affects the future value. This approach is also widely applicable to a number of different domains where there is a combinatorial problem embedded in a DP.

The particular problem we will focus to highlight the relevance and efficacy of this approach is from a collaboration with a start-up in the on demand health care space. They aim to provide health care delivered to the home. They provide an app based system that allows a user to request a nurse practitioner to treat them at their own house. A nurse practitioner is a qualified health care provider, licensed to prescribe medication. They primarily treat common illnesses, which are the cases the start up provides service for. Our focus is on routing the nurse practitioners efficiently. This means matching patients with available nurse practitioners and scheduling patients with nurse practitioners when the system is busy. They have a goal to serve all patients within a certain time frame, in this case, one hour. This is a constraint the solution must respect. If a patient is not able to be seen within this time span, they are referred to their local urgent care. In addition, the tools developed are very useful for planning purposes as it allows the company to simulate different demand scenarios and staffing decisions and explore the potential effects on their level of service.

The company aims to address several issues with the current practice of how low acuity health care is provided, including:

- **High costs of running a brick and mortar clinic**: An average Emergency Department visit costs $1230, while an Urgent Care visit costs $245 - $500. For each the average facility fee accounts for approximately 50% (Caldwell et al.

(2013)) of the cost of service. In both cases, the high costs involved with running and maintaining a brick and mortar clinic suggest that it might be possible to deliver cheaper health care options.

- **Inefficiencies in seeing patients**: Current static models of health care may result in imbalances in matching supply and demand. A common issue faced by brick and mortar health care providers is that some clinics receive very high demand and long wait times, while other locations have low demand resulting in nurse practitioners being idle. By making health care providers mobile, it may be possible to overcome some of these inefficiencies.

- **Poor customer satisfaction**: Feedback from market research conducted by our industry collaborator indicates that patients are currently dissatisfied with the current status quo in having health care providers at fixed locations. There is significant interest and opportunity in having in home services from a convenience stand point.

Devising efficient and practical routing techniques will help overcome some of these issues as it allows an on-demand home delivery system to be feasible and economical.

4.2 Relevant literature

Our work lies at the intersection of three streams of literature: dynamic vehicle routing, approximate dynamic programming and home health care scheduling.

4.2.1 Dynamic vehicle routing

Dynamic vehicle routing typically involves finding heuristic strategies to route vehicles which adapt in response to new requests for service (Gendreau et al. (1995),Cordeau et al. (2002), Bent and Van Hentenryck (2007)). The literature has explored a wide range of techniques from the area of stochastic optimization (see for example, Bent and Van Hentenryck (2004a)) and metaheuristics (see for example, Gendreau et al.

(1999)). Our problem can be thought of as a special case of the dynamic vehicle routing problem with time windows (Gendreau et al. (1999), Kohl et al. (1999), Chen and Xu (2006)). Since there is a service goal of serving all patients within an hour window, the time window begins immediately when the patient arrives and waiting ends after an hour. This means that patient arrivals are not known in advance, which makes the problem challenging. We focus on using an approximate dynamic programming approach to solve this problem. For a comprehensive review of dynamic vehicle routing, see Pillac et al. (2011).

4.2.2 Approximate dynamic programming

In vehicle routing problems the resulting dynamic program (DP) is very complex with a large state space. Due to the curse of dimensionality the DP formulation can't be solved exactly. Approximate dynamic programming techniques are utilized and aim to approximate the cost-to-go function function using a lower dimensional basis and function approximations. A range of approximations have been explored in the literature including linear (Powell and Carvalho (1998), De Farias and Van Roy (2003)), separable concave (Topaloglu and Powell (2003)), non-parametric kernel (Bhat et al. (2012)) or neural networks (Mnih et al. (2013), Mnih et al. (2015)). A number of these techniques have also been applied to a vehicle routing setting (Powell and Topaloglu (2003), Topaloglu and Powell (2006), Novoa and Storer (2009), Powell et al. (2007a), Godfrey and Powell (2002b)). In contrast, we explore tree or tree ensemble approximations of the cost-to-go function while capturing the specifics of the nurse practitioner routing setting. We expand on this comparison with the current literature in Section 4.5. We refer the reader to Bertsekas (1995) and Powell (2007) for a complete review of approximate dynamic programming techniques.

4.2.3 Home health care scheduling

There is a significant literature on home health-care routing. These papers are predominantly static and deterministic, where the patient arrivals are well known in

advance (Yalçındağ et al. (2012), Torres-Ramos et al. (2014), Morrice et al. (2014), Koeleman et al. (2012), Lanzarone et al. (2012)). This is a reasonable assumption in many cases, such as with outpatients at a hospital where patients are scheduled to be treated several times to provide treatment. There has been some work to introduce some stochasticity into travel time and demand (Lanzarone and Matta (2014), Yuan et al. (2015)), and incorporate dynamic patient arrivals into an existing schedule (Bennett and Erera (2011), Demirbilek et al. (2019)). The setting we study is an on-demand model of health care delivery, with patients aimed to be served within an hour of requesting service. As such, the patients which will be served throughout the day are not known in advance. This setting is significantly more dynamic and as a result models previously studied do not apply. Furthermore, we believe we are the first to use approximate dynamic programming methods in this problem area. There are papers from the operations management literature which address dynamic scheduling (Patrick et al. (2008); Santibáñez et al. (2007); Levi et al. (2007); Begen and Queyranne (2011)), but do not take into account the vehicle routing component. For a comprehensive review on the home health care literature we refer the reader to the review by (Fikar and Hirsch (2017)).

4.3 Contributions

In what follows, we summarize the overall contributions of this work.

- Model the nurse practitioner routing problem using Markov Decision Process and solve it using Approximate Dynamic Programming.

- Formulate general ADPs with combinatorial action spaces and non-linear cost-to-go functions using a tree or tree ensemble approximation.

- Present a novel, Mixed Integer Optimization formulation for optimizing over a tree. We prove this formulation is ideal. We extend to include assignment constraints and binary inputs and explore properties of these formulations.

- Using extensive numerical experiments, we show that these formulations are able to outperform other formulations in terms of speed to find the optimal solution.

- Show that the tree approximation performs well relative to other ADP approximations for the nurse practitioner problem using numerical experiments.

4.4 Model

The nurse practitioner routing problem can be formulated as a Markov Decision Process (MDP). This formulation is defined by the state, action set, reward function and transition function.

State:

The geographic area being served can be discretized into D locations. Define a binary vector $l_{jt}^P \in \{0,1\}$ which is equal to 1 if there is a patient at location j at time t and 0 otherwise, while $l_{it}^N \in \{0,1\}$ is equal to 1 if there is a nurse practitioner at location i at time t. In line with the company's commitment to serve each patient within a time span h, assumed to be an hour, a patient has a time remaining $\tau_{kt} \in (0,h)$ they need to be served by, set to a default of 0 if there is no patient at that location. Each nurse practitioner has a time $a_{kt} \in \mathbb{R}_+$ at which they will be available, with a default of 0 if there is no nurse practitioner at that location. This is typically when they finish serving the current patient. The service time for a patient is defined as μ and is assumed to be constant. We denote the state of the system as $\mathcal{S}_t = (\boldsymbol{l_t^P}, \boldsymbol{\tau_t}, \boldsymbol{l_t^N}, \boldsymbol{a_t})$.

Action set: The set of feasible actions that can be taken is defined by the polyhedral set $\mathcal{A}$, where x_{ij} is a binary decision variable which is equal to one if nurse practitioner in location i is assigned to patient in location j and is zero otherwise.

$$\mathcal{A}_t \;=\; \{\boldsymbol{x} | \sum_{j=1}^{D} x_{ij} \leq l_{jt}^P \qquad \forall j \in \{1,...,D\}, \tag{4.1a}$$

$$\sum_{j=1}^{D} x_{ij} \leq l_{it}^{N} \qquad \forall i \in \{1, ..., n_t\}, \tag{4.1b}$$

$$\sum_{i=1}^{D} (d_{ij} + a_{it}) x_{ij} \leq \tau_{jt} \qquad \forall j \in \{1, ..., D\}, \tag{4.1c}$$

$$x \in \{0, 1\} \qquad \forall i \in \{1, ..., D\},\ \forall j \in \{1, ..., D\}\ \} \tag{4.1d}$$

Constraint (4.1b) requires that each nurse practitioner is used only once within each time period. This is sufficient in this setting because the time between the dispatch for the current patient and the start of the next patient's service (the time taken for two journeys and a service), is larger than the maximum wait time for each patient. Constraint (4.1a) requires that each patient is served at most once. The time taken to travel between the locations of a nurse practitioner and patient is defined as d_{ij}. Constraint (4.1c) requires each patient be served on time, since $x_{ij} = 1$ implies $d_{ij} + a_{it} \leq \tau_{jt}$. The allocations x_{ij} for which this is not feasible can also be pruned prior to solving.

Reward: A general reward associated with assigning nurse practitioner i to patient j can be defined as r_{ij}. This function can capture various objectives of interest to the health care provider. In high demand settings, it is desirable that the number of patients seen is maximized. This can be achieved by setting $r_{ij} = 1$. In lower demand settings, there may be many allocations which can serve all patients. In this setting, it may be desirable to penalize the distance the nurse practitioners travel, by using a reward $r_{ij} = \theta - d_{ij}$. In this case, θ is a sufficiently large number that has a goal to prioritize seeing as many patients as possible over short distances. In Section 4.7, we explore these objectives. Similarly, it is also possible to minimize patient wait times with the reward $r_{ij} = \theta - (\tau_{jt} - (d_{ij} + a_{it})$. It is also possible to incorporate multiple objectives through appropriate weighting of the aforementioned functions or more sophisticated multi-objective optimization approaches.

System dynamics: Define $\tilde{t}$ as the time of the next patient arrival. Define a transition function from one state to the next $(\boldsymbol{l}_{\tilde{\boldsymbol{t}}}^{\boldsymbol{P}}, \boldsymbol{\tau}_{\tilde{\boldsymbol{t}}}, \boldsymbol{l}_{\tilde{\boldsymbol{t}}}^{\boldsymbol{N}}, \boldsymbol{a}_{\tilde{\boldsymbol{t}}}, \boldsymbol{x}) = f(\boldsymbol{l}_{\boldsymbol{t}}^{\boldsymbol{P}}, \boldsymbol{\tau}_{\boldsymbol{t}}, \boldsymbol{l}_{\boldsymbol{t}}^{\boldsymbol{N}}, \boldsymbol{a}_{\boldsymbol{t}}, \boldsymbol{x})$. The transition depends on whether the next patient arrives before or after the nurse

practitioners have finished serving their current patient. If a nurse practitioner is assigned to a patient $x_{ij} = 1$, and the nurse practitioner is available by the time the next patient arrives, $\tilde{t} \geq a_{it}$ then dispatch will occur and the nurse practitioner's location is updated to the patient's location $l^N_{i\tilde{t}} = l^P_{jt}$. The time the nurse practitioner is next available is updated to $a_{i\tilde{t}} = \max(t, a_{it}) + d_{ij} + \mu$, which is the time it will take to travel to and serve the next patient. The patients for which a nurse practitioner has been dispatched are removed from the set of patients that need to be seen. We assume that the nurse practitioners will stay stationary after treatment. This allows them time to fill out paperwork related to the previous case and prevents the accumulation of unnecessary mileage. This is also inline with the nurse practitioners' preferences when surveyed. If the next patient arrives before the nurse practitioner is available, $\tilde{t} < a_{it}$ then no nurse practitioner is dispatched, the location remains the same $l^N_{i\tilde{t}} = l^N_{it},\ l^P_{j\tilde{t}} = l^P_{jt}$ and time advances $a_{i\tilde{t}} = a_{it} - (\tilde{t} - t),\ \tau_{j\tilde{t}} = \tau_{jt} - (\tilde{t} - t)$. Patient arrivals follow a memoryless distribution, such as a Poisson process, with rate λ. The new patient has a random location according to a disctrete random variable $l^P_{(m_t+1)\tilde{t}} = U_{\tilde{t}}$, while the time at which the patient needs to be seen is set to $\tau_{(m_t+1)\tilde{t}} = h$. It is assumed the number of locations is sufficiently large that the probability of multiple patients at the same location is negligible, although the formulation can be modified to take that into account.

Our goal in the nurse practitioner routing problem is to maximize the reward over a finite time horizon T, although the methods presented can also be adapted to the infinite horizon setting. This Markov Decision Process can theoretically be solved using dynamic programming, through the formulation of Bellman's equation:

$$J(\mathcal{S}_t) = \max_{\boldsymbol{x} \in A_t(S_t)} \boldsymbol{r}(\mathcal{S}_t)\boldsymbol{x} + E[J(f(\mathcal{S}_t, \boldsymbol{x}))] \tag{4.2}$$

This method is known to balance the reward from the first stage, with the expected reward from future stages. However, due to the curse of dimensionality, solving this MDP using dynamic programming methods is not tractable and as a result, approximate methods must be used.

Furthermore, the optimization sub-problem to be solved for each state and stage of a backwards recursion policy (finite state) value iteration policy (infinite state) is not tractable. This is because the term $E[J(f(\mathcal{S}_t, \boldsymbol{x}))]$ is non-linear. This means we can't exploit the favorable structural properties of the linear portion of the problem $\underset{\boldsymbol{x} \in A_t(S_t)}{max} \boldsymbol{r}(\mathcal{S}_t)\boldsymbol{x}$, which is otherwise efficient to solve. In general, the approach used to solve these problems is enumeration (see for example, Silver et al. (2016), Mnih et al. (2015)). If we define $\boldsymbol{x^i}$ to be one of k possible feasible solutions, the problem is solved as:

$$\underset{\boldsymbol{x^i} \in \{\boldsymbol{x^1}, \ldots, \boldsymbol{x^k}\}}{max} \boldsymbol{r}(\mathcal{S}_t)\boldsymbol{x^i} + E[J(f(\mathcal{S}_t, \boldsymbol{x^i}))] \tag{4.3}$$

This solution method faces a number of practical issues, primarily that the number of different solutions $\boldsymbol{x^i}$ is exponential in the number of nurse practitioners and patients considered. As a result this approach is intractable for this problem and many others which have combinatorial subproblems. This is compounded by the iterative nature of value or policy iteration, which results in this optimization problem having to be solved a very large number of times. In these settings, being able to solve this optimization problem quickly is important due to the sheer number of times it is solved and due to the company needing to get real time solutions. To address this issue, we use approximate dynamic programming combined with machine learning. These methods work by approximating the state space and cost-go-go function.

Extensions: It is also possible to add constraints that capture useful practical variations to this model:

- **Continued service:** Can place restrictions so that once the patient is guaranteed service, they will not be left out of a future allocation. This can be achieved by setting $\sum_{j=1}^{D} x_{ij} = 1$ for these patients.

- **Extension of maximum wait time:** With the model as described, each nurse practitioner has no more than one patient in the queue, due to the relatively tight maximum waiting time guarantees the company provides. This can be extended to having up to two patients in the queue using variables x_{ijk} to

denote if nurse practitioner i serves patient j then k, along with appropriate 3-way matching and feasibility constraints. While from limited experimentation this still solves quickly using the approaches discussed in this chapter, extending to three or more is likely to considerably affect solution time.

- **Incorporating on-call nurse practitioners:** It is also possible to incorporate different employment models, such as having on-call nurse practitioners available at a fixed cost per service in addition to full time employees whose wage is assumed to be a fixed cost. This can be modeled by having two classes of nurse practitioners, one of which has an additional penalty for each patient seen. This is useful for the company in planning employment decisions.

4.5 Approximate methods

The state space of the nurse practitioner routing problem is too large for traditional dynamic programming approaches to be tractable, as the number of different combinations of nurse practitioner, patient locations, and availability times is exponential with the problem size. As a result, approximate dynamic programming approaches must be used, where the problem is approximated using a reduced state space and an approximate value function. The policies obtained are not necessarily optimal, but often perform well in practice.

Basis function approximation: In the nurse practitioner setting, an accurate approximation of the expected future reward can be made using a significantly lower dimensional state space. We propose using only the location of the nurses as the reduced state space. This reduces the size of the problem considerably and makes the dynamic programming algorithm more tractable. Furthermore, this particular basis function *linearizes* the transition function. This is important in the context of the subproblem that needs to be solved as part of the dynamic programming algorithm — it allows this subproblem to be solved as a linear optimization problem (with integer variables), for which there are many fast solvers available for solving problems of a reasonable size.

We approximate $J(\mathcal{S}_t)$ with $F(\Phi(\mathcal{S}_t))$ where $\Phi(\cdot) : \{0,1\}^D \times (0,h)^D \times \{0,1\}^D \times \mathbb{R}_+^D \to \{0,1\}^D$ is a basis function reducing the state space to a binary representation of the location of each nurse practitioner after each patient has been served, with a 1 in position j if there is a nurse practitioner present. This captures the intuition that the distribution of nurse practitioners, and how well it matches future demand, is an important driver of future profit. In our case the basis approximation of the future state can be approximated as a linear function of the decision :

$$\Phi(f(\mathcal{S}_t, \boldsymbol{x}))_i = l_{it}^N - \sum_{j=1}^{D} x_{ij} + \sum_{j=1}^{D} x_{ji} \tag{4.4}$$

If we define $w_i = \Phi(f(\mathcal{S}_t, \boldsymbol{x}))_i$ as the decision variable of whether there is a nurse practitioner in location i in the future, this results in the following optimization problem:

$$\max_{\boldsymbol{x}} \sum_{i=1}^{D} \sum_{j=1}^{D} r_{ij} x_{ij} + F(\boldsymbol{w}) \tag{4.5a}$$

$$\text{s.t.} \quad \boldsymbol{x} \in \mathcal{A} \tag{4.5b}$$

$$w_i = l_{it}^N - \sum_{j=1}^{D} x_{ij} + \sum_{j=1}^{D} x_{ji} \qquad \forall i \in \{1, ..., D\} \tag{4.5c}$$

We also experiment with other basis functions including, $\Phi(l_{jt}^P, \tau_{jt}, l_{it}^N, a_{it}) = a_{it}$ a mapping of the entire state to one where we just have when the nurse practitioners will be available after the assigned patients are served, with a default high value if there is no nurse practitioner at that location. This captures the intuition that the profit is not only based on where nurse practitioners are located, but also when they will become available. This can also be written as a linear function, but for simplicity of exposition, we use (4.4) in all future formulations.

Value function approximation: Despite the basis function approximation reducing the size of the problem, the state space is still too large to store the cost-to-go

function in a look up table. The number of different combinations of nurse practitioner locations is still significant. This motivates the use of an approximate value functions, which is able to be trained from data on previous states encountered and the corresponding reward which was obtained.

$F(\cdot) : \{0,1\}^D \rightarrow \mathbb{R}$ is a function which maps the state to the predicted reward. Several different approximations have been proposed in the approximate dynamic programming literature:

- Linear: $F(\Phi(\mathcal{S}_t)) = \sum_i \theta_i \Phi(\mathcal{S}_t)_i$ (Powell and Carvalho (1998))
- Separable, concave functions: $F(\Phi(\mathcal{S}_t)) = \sum_i f_i(\Phi(\mathcal{S}_t)_i)$, where $f_i(\cdot)$ is a concave (often piece-wise linear) function (Topaloglu and Powell (2003)).
- Non-parametric machine learning functions such as neural networks $F(\Phi(\mathcal{S}_t)) = \tilde{F}(\Phi(\mathcal{S}_t))$ (Mnih et al. (2013)).

Linear functions are advantageous because the optimization problem becomes linear (or at least inherits the complexity of the DP optimization subproblem), resulting in a tractable optimization problem. In the setting where there is a coarse discretization of locations, it is possible to have multiple nurse practitioners in the same location. In this setting it is desirable to capture the diminishing returns that occurs from having multiple nurse practitioners in the same region. This motivates the use of separable functions , and in particular concave, piece-wise linear functions which also lead to tractable optimization problems. Topaloglu and Powell (2003) present algorithms to estimate these functions so that the resulting formulations have favorable properties for solving the subproblem. Separable functions have the drawback that they do not capture the interaction effects between nurse practitioners in adjacent locations, which are able to serve patients that arise at nearby demand nodes. This captures the notion that it is desirable to have the nurse practitioners dispersed around the region served. Furthermore, another issue with separable functions is that they are often not accurate in terms of predicting the cost-to-go function, due to the fact that it is inherently non-linear. On the other hand, non-parametric machine

learning functions such as neural networks have high predictive accuracy, but often it is not known how to optimize over the function, treating it as a black box or using enumeration methods. This problem also exists for look-up-tables in a traditional DP approach. We propose an approach which uses trees or tree ensembles to approximate the cost-to-go function. The advantage in using trees instead of tree ensembles is that optimizing over a single tree is much faster. Apart from these tree based methods having a high accuracy, it was shown in chapter 3 that they can be modeled using MIP techniques and efficiently solved using modern optimization algorithms.

Estimation: We follow a data-driven approach, using data collected from an existing policy to improve system performance. To achieve this we use a step of direct policy iteration (Lagoudakis and Parr (2002)), alternatively known as monte carlo learning (Roy and McCallum (2001)). If we denote sample path k as a sequence of states $\mathcal{S}_{kt_1}, ..., \mathcal{S}_{kt_n}$, we can find the cumulative reward associated with a sample path within the time horizon $R_k^\pi = \sum_i (\sum_{t'|t<t'<t+T} r(\mathcal{S}'_{kt})\pi(\mathcal{S}_{t'})$, which is a noisy estimate of the (unknown) cost to go function associated with the policy the routing algorithm was using $J^\pi(\mathcal{S}_{kt})$. This can be considered as data for the problem. We would like our function estimate to fit the sampled cumulative reward well, resulting in the following estimation problem to find the approximate cost-to-go function $F^\pi(\cdot)$:

$$\min_{F^\pi(\cdot)} \frac{1}{n} \sum_{k=1}^{n} (R_k^\pi - F(\Phi(\mathcal{S}_{tk})))^2 \tag{4.6}$$

Further rounds of policy iteration can be implemented using the following algorithm:

Algorithm 5 Policy iteration

1: *Generate data R_k^π form a policy π by solving formulation (4.5) over a range of randomly generated patient arrivals.*

2: *Estimate $F^\pi(\cdot)$ by solving estimation procedure (4.6)*

3: *Repeat steps 1. and 2. until convergence*

However, in our simulations the first order difficulty is finding accurate approximations of $J^\pi(\mathcal{S}_t)$, due to the fact that the estimation is not very sensitive to the

actual policy that was simulated (within reason). This reflects a) that this problem is very noisy and b) the difference in performance of the different policies is small relative to the magnitude of the noise. We explore policy iteration in for this problem in section 4.7.2.

If further rounds of policy iteration are used, it is useful to consider how the trees are updated from round to round. It is common in reinforcement learning to have batched updates of the data, so that at each round the function is trained using new data in addition to older data from previous rounds Lange et al. (2012). As it can be time consuming to generate enough data to get good estimations for each policy, an approach which updates the policy more often has been shown to work well. In this setting, local search methods from Bertsimas and Dunn (2017) can be used to quickly re-optimize the tree as new data comes available.

4.5.1 General formulation

The methods developed in this chapter apply more generally to optimization problems where we have a polyhedral set $\mathcal{A} = \{(x, u)|Ax + Bu \leq b, x \in \mathbb{R}^d, u \in \mathbb{Z}\}$ possibly including integer variables, a linear function $w = Cx + Du$ linking the decisions to the future state approximation, and a cost-to-go function which can be approximated with a tree or a tree ensemble $F(\cdot)$:

$$\max_{x,u} \quad c'x + d'u + F(w) \tag{4.7a}$$

$$\text{s.t.} \quad Ax + Bu \leq b \tag{4.7b}$$

$$Cx + Du = w \tag{4.7c}$$

$$x \in \mathbb{R}^d, u \in \mathbb{Z} \tag{4.7d}$$

We also note that although we formulated this problem as a Markov Decision Problem, we can also consider this a largely 'model free' formulation, in the sense that we use minimal information about the system, and use data and non-parametric function approximation to learn how to behave.

4.5.2 Myopic LP

This Section outlines an approach that we benchmark against in numerical experiments in Section 4.7 in addition to the approximate dynamic programming policies outlined in Section 4.5.

A heuristic approach which can be used to solve this optimization problem is to set $E[J(f(\mathcal{S}_t, \boldsymbol{x}))] = 0$, therefore ignoring the impact of decisions on the future profit. This results in solving an LP at every stage:

$$\max_{\boldsymbol{x}} \sum_{i=1}^{D} \sum_{j=1}^{D} r_{ij} x_{ij} \tag{4.8a}$$

$$\text{s.t.} \quad \boldsymbol{x} \in \mathcal{A} \tag{4.8b}$$

This is solved in a reoptimization framework; every time a new patient arrives into the system, the allocation is reoptimized to ensure that the current approach is optimal. The optimal solution serves as a plan for how patients will be served as nurse practitioners come available. Once a nurse practitioner becomes available and is dispatched to see the patient, that pair in the allocation is considered as fixed and will not change in future reoptimization. Before the next optimization routine is run, the parameters a_i and τ_j are updated accordingly.

Using an optimization algorithm has the largest impact if the demand is high, and there are a large number of patients that are awaiting service. In this regime, market thickening behavior can occur, making it more likely that there will be an assignment of patients to nurse practitioners which is more beneficial than the approach of assigning the nurse practitioner who will arrive first to a patient.

4.6 Mixed Integer Optimization formulations

In this section we are concerned with solving the optimization of the nurse practitioner allocation problem, where the cost-to-go is approximated using a tree:

$$\max_{\boldsymbol{x}} \sum_{i=1}^{D}\sum_{j=1}^{D} r_{ij}x_{ij} + y \tag{4.9a}$$

$$\text{s.t. } \boldsymbol{x} \in \mathcal{A} \tag{4.9b}$$

$$\boldsymbol{w}, y \in \mathcal{T} \tag{4.9c}$$

$$w_i = l_{it}^N - \sum_{j=1}^{D} x_{ij} + \sum_{j=1}^{D} x_{ji} \qquad \forall i \in \{1, ..., D\} \tag{4.9d}$$

We represent the tree as a polytope $\mathcal{T}$, which enforces the relationship between the future location of nurse practitioners $\boldsymbol{w}$ and the approximate cost-to-go function y. We begin this section with a general MIP formulation for a tree $\mathcal{T}$. We then prove some favorable properties of this formulation, such as showing that it is ideal, and that the constraints are facet defining. We then focus on a formulation for when $\boldsymbol{w}$ is binary, as is the case for the nurse practitioner allocation problem. We also prove that this formulation is ideal, both in terms of the binary variables introduced to model the tree, and the variables $\boldsymbol{w}$. Finally we explore the interaction between the tree polytope and the assignment polytope, and show how to introduce cutting planes which strengthen the resulting formulation. The tight formulations explored in this chapter allow for the subproblems at each stage of a dynamic programming to be solved efficiently, therefore allowing the entire dynamic program to be solved tractably.

4.6.1 MIO formulation for a single tree

Each leaf $l \in \{1, ..., p\}$ in a standard axis aligned tree is a hyperrectanglular set $\mathcal{L}_l$, defined by an upper u_{il} and a lower (bottom) b_{il} bound for each feature w_i. Each leaf is also associated with an estimate (or prediction) s_l of the outcome associated with that leaf. Throughout, we assume w_i is bounded.

Definition 3. *A leaf is defined as:*

$$\mathcal{L}_l = \{\boldsymbol{w}, y \mid w_i \leq u_{il} \qquad \forall\ i \in \ \{1...d\}, \tag{4.10a}$$

$$w_i \geq b_{il} \qquad i \in \{1...d\}, \tag{4.10b}$$

$$y = s_l\} \tag{4.10c}$$

The upper and lower bounds can be calculated from the splits in the tree with $b_{li} = \max s_{li} | s_{li} \in G_{li}$, where G_{li} is the set of "greater than" splits which lead to leaf l for feature i, while $u_{li} = \min s_{li} | s_{li} \in L_{li}$, where L_{li} is the set of "less than splits" which lead to leaf l for feature i.

Union of polyhedron formulation

We can formulate a tree as a union of polyhedra:

$$(\boldsymbol{w}, y) \ \in \ \bigcup_{l=1}^{p} \mathcal{L}_l = \mathcal{T} \tag{4.11}$$

This can be achieved using the classical extended formulation from Jeroslow (1987), which introduces many auxiliary variables to model the set. This is also known as a "multiple choice" formulation.

$$\mathcal{T}^{\text{ext}} = \{\boldsymbol{w}, y | \ u_{li} z_l \geq w_i^l \qquad \forall i \in \{1...d\},\ \forall l \in \{1, ..., p\} \tag{4.12a}$$

$$b_{li} z_l \leq w_i^l \qquad \forall i \in \{1...d\},\ \forall l \in \{1, ..., p\} \tag{4.12b}$$

$$y^l = s_l z_l, \qquad \forall l \in \{1, ..., p\} \tag{4.12c}$$

$$\sum_{l=1}^{p} z_l = 1, \tag{4.12d}$$

$$w_i = \sum_{l=1}^{p} w_i^l \qquad \forall i \in \{1...d\} \tag{4.12e}$$

$$y = \sum_{l=1}^{p} y^l, \qquad \forall l \in \{1, ..., p\} \tag{4.12f}$$

$$z_l \in \{0, 1\} \qquad \forall l \in \{1, ..., p\}\} \tag{4.12g}$$

In this formulation there are p variables $\boldsymbol{w}^l$, corresponding to each leaf of the tree. Here, z_l is a binary variable, equal to 1 if $\boldsymbol{w}$ is in leaf l and 0 otherwise. This formulation is ideal as proved in Jeroslow and Lowe (1984) and Balas (1985), but often has computational issues when solved in practice. Not only does it introduce a large number of auxiliary variables, but it is well known in the mixed integer optimization community that these formulations suffer from degeneracy, resulting in poor performance in practice (Vielma (2018)).

Definition 4. *A MIP formulation $Ax + By \leq b, x \in \mathbb{R}^n, y \in \mathbb{Z}^m$ is ideal if and only if its LP relaxation has at least one basic feasible solution and all the extreme points of the LP relaxation are integer $y \in \mathbb{Z}^m$.*

We improve upon this formulation, by projecting on to $\boldsymbol{w}$. This eliminates the variables $\boldsymbol{w}^l$ and thus results in a significantly smaller formulation. Furthermore, we can prove this formulation is ideal for a single tree.

$$\mathcal{T}^{\text{proj}} = \{\boldsymbol{w}, y| \sum_{l=1}^{p} u_{li} z_l \geq w_i \qquad \forall i \in \{1...d\}, \tag{4.13a}$$

$$\sum_{l=1}^{p} b_{li} z_l \leq w_i \qquad \forall i \in i \in \{1...d\}, \tag{4.13b}$$

$$y = \sum_{l=1}^{p} z_l s_l \tag{4.13c}$$

$$\sum_{l=1}^{p} z_l = 1, \tag{4.13d}$$

$$z_l \in \{0,1\} \qquad \forall l \in \{1,...,p\}\} \tag{4.13e}$$

The main idea behind this proof is that the union of polyhedra formulation (4.12) is ideal, and therefore the projection onto variables $\boldsymbol{w}$ is also ideal. These ideal projected formulations always exist, but in general the projection is not a tractable operation, and can result in a formulation with exponentially many constraints. In this special case, the resulting formulation (4.13) only has $2d+1$ constraints (in addition to binary constraints) and $p + d$ variables. This has the benefit over formulation 4.12 that it

has significantly fewer variables and less degeneracy.

Theorem 5 (Ideal formulation for a tree)**.** *The polyhedra $\mathcal{T}^{proj}$ is ideal.*

Proof. We prove this by applying Fourier-Motzkin elimination to formulation (4.12) to eliminate all $\boldsymbol{w}^l$, and showing we arrive at formulation (4.13). An overview of the technique can be found in Hooker (2011).

Let us begin by applying Fourier-Motzkin elimination to eliminate $\boldsymbol{w}^1$. This results in the following constraints:

$$b_{1i}z_1 \leq u_{1i}z_1 \qquad \forall i \in \{1...d\} \tag{4.14a}$$

$$b_{1i}z_1 \leq w_i - \sum_{l\in\{2,...,p\}} w_i^l \qquad \forall i \in \{1...d\} \tag{4.14b}$$

$$u_{1i}z_1 \geq w_i - \sum_{l\in\{2,...,p\}} w_i^l \qquad \forall i \in \{1...d\} \tag{4.14c}$$

$$u_{li}z_l \geq w_i^l \qquad \forall i \in \{1...d\},\ \forall l \in \{2,...,p\} \tag{4.14d}$$

$$b_{li}z_l \leq w_i^l \qquad \forall i \in i \in \{1...d\},\ \forall l \in \{2,...,p\} \tag{4.14e}$$

$$y^l = s_l z_l, \forall l \in \{1,...,p\} \tag{4.14f}$$

$$y^l = \sum_{l=1}^{p} y_i^l, \qquad \forall l \in \{1,...,p\} \tag{4.14g}$$

$$\sum_{l=1}^{p} z_l = 1, \tag{4.14h}$$

$$z_l \in \{0,1\} \qquad \forall l \in \{2,...,p\} \tag{4.14i}$$

Constraint (4.15a) is redundant since $b_{l1} \leq u_{l1}$ by definition, and can therefore be eliminated. We can use another round of elimination to remove $\boldsymbol{w}^2$. This results in the following constraints:

$$b_{2i}z_2 \leq u_{2i}z_2 \qquad \forall i \in \{1...d\} \tag{4.15a}$$

$$b_{1i}z_1 + b_{2i}z_2 \leq w_i - \sum_{l\in\{3,...,p\}} w_i^l \qquad \forall i \in \{1...d\} \tag{4.15b}$$

$$u_{1i}z_1 + u_{2i}z_2 \geq w_i - \sum_{l\in\{3,...,p\}} w_i^l \qquad \forall i \in \{1...d\} \tag{4.15c}$$

$$u_{li}z_l \geq w_i^l \qquad \forall i \in \{1...d\},\ \forall l \in \{3,...,p\} \tag{4.15d}$$

$$b_{li}z_l \leq w_i^l \qquad \forall i \in i \in \{1...d\},\ \forall l \in \{3,...,p\} \tag{4.15e}$$

$$y^l = s_l z_l, \qquad \forall l \in \{1,...,p\} \tag{4.15f}$$

$$y^l = \sum_{l=1}^{p} y_i^l, \qquad \forall l \in \{1,...,p\} \tag{4.15g}$$

$$\sum_{l=1}^{p} z_l = 1, \tag{4.15h}$$

$$z_l \in \{0,1\} \qquad \forall l \in \{3,...,p\} \tag{4.15i}$$

If this process is repeated iteratively to eliminate $\boldsymbol{w}^3, ..., \boldsymbol{w}^p$, and y^l is eliminated by simple substitution (for $z_l s_l$), we arrive at formulation (4.13). □

If formulation (4.13) is reformulated slightly, we can prove some additional favorable properties of this formulation. One of the variables z_p can be eliminated through the substitution $z_p = 1 - \sum_{l=1}^{p-1} z_l$. Consequently, $\boldsymbol{z} \in \{0,1\}^{p-1}$ and as a result, $\boldsymbol{z} = 0$ implies $\boldsymbol{w} \in \mathcal{L}_p$.

$$\mathcal{T}^{\text{proj}} = \{\boldsymbol{z}, \boldsymbol{w}|\ u_{pi} + \sum_{l=1}^{p-1}(u_{li} - u_{pi})z_l \geq w_i \qquad \forall i \in \{1...d\}, \tag{4.16a}$$

$$b_{pi} + \sum_{l=1}^{p-1}(b_{li} - b_{pi})z_l \leq w_i \qquad \forall i \in i \in \{1...d\}, \tag{4.16b}$$

$$y = s_p + \sum_{l=1}^{p-1} z_l(s_l - s_p) \tag{4.16c}$$

$$z_l \in \{0,1\} \qquad \forall l \in \{1,...,p-1\}\} \tag{4.16d}$$

In particular, we can show that under mild assumptions, (4.16a) and (4.16b) are facet defining. The proof technique is similar to that in Anderson et al. (2018).

Proposition 3. *For all $l \in \{1...p\}$, assume $\mathcal{L}_l$ is non-empty. Furthermore, assume*

that for some $k \in \{1...p\}$, $\mathcal{L}_k$ is full dimensional (has dimension d). Then constraints (4.16a) and (4.16b) are facet defining for leaf k.

Proof. We show that (4.16b) is facet defining. It can be proved that (4.16a) is facet defining using the same argument. The dimension of this polyhedron is $p + d$. To show constraint (4.16b) is facet defining, we need to find $p + d$ affinely independent points which satisfy $\sum_{l=1}^{p} u_{ni} + (u_{li} - u_{ni})z_l = w_i$.

Constraint (4.16b) places bounds on a dimension i of $\boldsymbol{w}$. Without loss of generality, consider $k = 1$ and $i = d$. Define $\hat{\boldsymbol{w}}$ as a point on the interior of the leaf $\mathcal{L}_1$ with respect to dimensions $1, ..., d-1$, but at the lower bound for dimension d, so that $\hat{w}_d = b_{1d}$. Define the point $\boldsymbol{q}^0 = (\hat{\boldsymbol{w}}, \boldsymbol{e}^1)$. This point satisfies (4.16b) with equality.

Consider $d-1$ points $\boldsymbol{q}^i = (\hat{\boldsymbol{w}} + \epsilon \boldsymbol{e}^i, \boldsymbol{e}^1)$ for $i \in \{1, ..., d-1\}$, where $\epsilon > 0$ is chosen to be sufficiently small that $\boldsymbol{q}^i \in \mathcal{L}_1$. A sufficiently small ϵ exists due to the fact that $\hat{\boldsymbol{w}}$ is on the interior with respect to dimensions $1, ..., d-1$. These points still satisfy (4.16b) with equality as q^i_d remains equal to $q^i_d = b_{1d}$.

Consider $p-1$ points $\tilde{\boldsymbol{q}}^i = (\tilde{\boldsymbol{w}}^i, \boldsymbol{e}^i)$ for $i \in \{2, ..., p-1\}$ and $\tilde{\boldsymbol{q}}^p = (\tilde{\boldsymbol{w}}^i, 0)$, where $\tilde{\boldsymbol{w}}^i \in \mathcal{L}_i$ and $\tilde{w}^i_i = l_{id}$. Such a point exists because $\mathcal{L}_i$ is non-empty.

We now need to show that these points are affinely independent. This can be proven by showing the following matrix is full rank:

$$\begin{pmatrix} \tilde{\boldsymbol{q}}^2 - \boldsymbol{q}^0 \\ \vdots \\ \tilde{\boldsymbol{q}}^p - \boldsymbol{q}^0 \\ \boldsymbol{q}^1 - \boldsymbol{q}^0 \\ \vdots \\ \boldsymbol{q}^{d-1} - \boldsymbol{q}^0 \end{pmatrix} = \begin{pmatrix} \tilde{\boldsymbol{w}}^2 - \hat{\boldsymbol{w}} & \boldsymbol{e}^2 - \boldsymbol{e}^1 \\ \vdots & \vdots \\ \tilde{\boldsymbol{w}}^p - \hat{\boldsymbol{w}} & \boldsymbol{0} - \boldsymbol{e}^1 \\ \epsilon \boldsymbol{e}^1 & \boldsymbol{0} \\ \vdots & \vdots \\ \epsilon \boldsymbol{e}^{d-1} & \boldsymbol{0} \end{pmatrix} \tag{4.17}$$

If we shift the $p-2$ last columns to be the the first $p-2$ columns and the $p-1$ to last column to $p-1$ from first, we end up with an upper diagonal matrix with non-zero entries on the diagonal, resulting in a matrix with full row rank. Since we only applied elementary operations to the original matrix, this is also has full row

rank.

$$\begin{pmatrix} \tilde{\boldsymbol{q}}^2 - \boldsymbol{q}^0 \\ \\ \vdots \\ \\ \\ \tilde{\boldsymbol{q}}^p - \boldsymbol{q}^0 \\ \boldsymbol{q}^1 - \boldsymbol{q}^0 \\ \vdots \\ \\ \boldsymbol{q}^{d-1} - \boldsymbol{q}^0 \end{pmatrix} = \begin{pmatrix} 1 & & & -1 & \tilde{w}_1^2 - \hat{w}_1 & \dots & \tilde{w}_{d-1}^2 - \hat{w}_{d-1} \\ & 1 & & -1 & & & \\ & & \ddots & \vdots & \vdots & \ddots & \\ & & & 1 \quad -1 & & & \\ & & & -1 & \tilde{w}_1^p - \hat{w}_1 & \dots & \tilde{w}_{d-1}^2 - \hat{w}_{d-1} \\ & & & & \epsilon & & \\ & & & & & \ddots & \\ & & & & & & \epsilon \end{pmatrix} \tag{4.18}$$

This proves that the points were affinely independent and (4.16b) is facet defining.

□

4.6.2 Ideal formulation for a single tree with binary features

In the nurse practitioner routing problem, the variables which are the input to the tree $\boldsymbol{w}$ represent whether there will be a nurse at location in the future. These variables are required to be binary. As such, it is desirable to find formulations where $\boldsymbol{w}$ will be naturally integer in addition to $\boldsymbol{z}$. We explore such formulations in this Section.

When the feature variables w_i are required to be binary, the bounds can be modified to $u_{li} = 1 \ \ \forall l \in H_i$, and 0 otherwise where $H_i = \{l | \ \forall \boldsymbol{w} \in \mathcal{L}_l, w_i = 1\}$, . Similarly, $b_{li} = 1 \ \ \forall l \in T_i$ and 0 otherwise, where $T_i = \{l | \ \exists \boldsymbol{w} \in \mathcal{L}_l, \ w_i = 1\}$. H_i can be interpreted as the set of leaves for which $w_i = 1$ is a parent node, while T_i can be interpreted as the set of leaves where $w_i = 0$ is not a parent node. Note that, $H_i \subseteq T_i$. This results in a modified formulation:

$$\mathcal{T}^{\text{binary}} = \{\boldsymbol{w}, y | \ \sum_{l \in H_i} z_l \leq w_i \qquad \forall i \in \{1...d\}, \tag{4.19a}$$

$$\sum_{l \in T_i} z_l \geq w_i \qquad \forall i \in i \in \{1...d\}, \tag{4.19b}$$

$$y = \sum_{l=1}^{p} z_l s_l \tag{4.19c}$$

$$\sum_{l=1}^{p} z_l = 1, \tag{4.19d}$$

$$z_l \in \{0,1\} \qquad \forall l \in \{1,...,p\}\} \tag{4.19e}$$

With these parameters, any feasible $\boldsymbol{w}$ will also be binary. We prove this result below:

Theorem 6 (Ideal formulation for a tree)**.** *The polyhedra $\mathcal{T}^{binary}$ is ideal.*

Proof. To prove this we will show by construction that for each basic feasible solution to the relaxation of (4.19), $\boldsymbol{z}, \boldsymbol{w}$ is binary. A basic feasible solution always has the equality constraints (4.19d and 4.19c) in the basis. Each basic feasible solution requires $p + d - 1$ out of the remaining $2d + p$ inequality constraints to be active, and for the set of active constraints to be independent.

Suppose (4.19a) and (4.19b) are both active for a particular index i. Then $\sum_{l \in H_i} z_l = \sum_{l \in T_i} z_l \implies \sum_{l \in T_i \setminus H_i} z_l = 0$, which sets some of the $z_l = 0$, which sets some of the variables $z_l = 0$. Let $a \leq d$ be the number of dimensions for which (4.19a) and (4.19b) are both active. This results in at least a variables $z_l = 0$, otherwise this subset of constraints is not linearly independent.

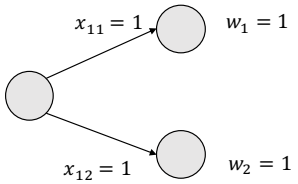
That leaves $p + d - 2a - 1$ constraints to be picked for the basis. There must be at least $d - a$ other dimensions for which one of (4.19a) or (4.19b) is active. It is not possible to have both inactive otherwise the resulting set of active constraints won't be linearly independent. This is because (4.19a) or (4.19b) are the only constraints where variable w_i appears.

Therefore, there are active $p - a - 1$ constraints to choose from the set $z_l \geq 0 \; \forall l \in \{1, ..., p\}$. This leaves a single leaf l^* for which z_l not set to zero, therefore $z_{l^*} = 1$ due to $\sum_{l=1}^{p} z_l = 1$. It follows that $w_i = 1$ for all i with $l^* \in H_i$, and $w_i = 0$ for all i with $l^* \notin T_i$. If $l^* \in T_i \setminus H_i$ and only 4.19b is in the basis, $w_i = 1$. If 4.19a is in the basis,

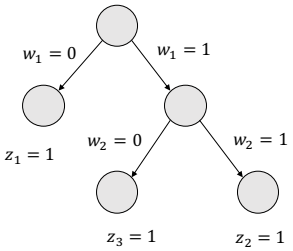
then $w_i = 0$. Therefore, for every active basis, $\boldsymbol{z}, \boldsymbol{w}$ is binary by construction. □

4.6.3 Combined tree and assignment polytope

It is a classical result that $\mathcal{A}_t$ is also ideal, as it has a totally unimodular matrix. One might hope that given that $\mathcal{A}$ and $\mathcal{T}^{\text{binary}}$ are ideal, that the intersection $\mathcal{T}^{\text{binary}} \cap \mathcal{A}$ is also ideal Unfortunately, there is an important negative result that the intersection is not generally ideal.



(a) Example nurse options



(b) Example tree

Figure 4-1: Example 1

Example 1 ($\mathcal{T}^{\text{binary}} \cap \mathcal{A}$ is not ideal)**.** *Suppose we have the following simple LP, with two customers and a single nurse practitioner:*

$$x_{11} + x_{12} \leq 1 \tag{4.20a}$$

$$w_1 = x_{11} \tag{4.20b}$$

$$w_2 = x_{12} \tag{4.20c}$$

$$z_2 \leq w_2 \tag{4.20d}$$

$$z_1 + z_2 \geq w_2 \tag{4.20e}$$

$$z_2 + z_3 \leq w_1 \tag{4.20f}$$

$$z_2 + z_3 \geq w_1 \tag{4.20g}$$

$$z_1 + z_2 + z_3 = 1 \tag{4.20h}$$

$$x_{11},\ x_{12},\ w_1,\ w_2,\ z_1,\ z_2,\ z_3 \geq 0 \tag{4.20i}$$

$x_{11} = 0.5, x_{12} = 0.5, w_1 = 0.5, w_2 = 0.5, z_1 = 0.5, z_2 = 0.5, z_3 = 0$ *is a basic feasible solution, which does not have binary variables.*

Even though this example shows that the formulation for the nurse practitioner routing problem is in general not ideal, the formulation can still be improved. We can strengthen this formulation by adding the following cuts. Define $H'_i = \{l|\ \forall \boldsymbol{w} \in \mathcal{L}_l \cap \mathcal{A}, w_i = 1\}$ as the set of *compulsory* leaves for a location, where the variable $\boldsymbol{w}$ additionally has to be within the $\mathcal{A}$. Similarly define, $T'_i = \{l|\ \exists \boldsymbol{w} \in \mathcal{L}_l \cap \mathcal{A}, w_i = 1\}$ and $N' = \{l|\ \nexists \boldsymbol{w} \in \mathcal{L}_l \cap \mathcal{A}\}$.

Proposition 4. *The following cuts are valid for formulation (4.19):*

$$\sum_{l \in H'_i} z_l \leq w_i \qquad \forall i \in \{1...d\}, \tag{4.21a}$$

$$\sum_{l \in T'_i} z_l \geq w_i \qquad \forall i \in \{1...d\}, \tag{4.21b}$$

$$z_l = 0 \qquad \forall l \in N' \tag{4.21c}$$

Proof. We show that for every $(\hat{\boldsymbol{x}}, \hat{w}, \hat{y}, \hat{z})$ which satisfies all constraints from formulations (4.21, 4.19, 4.9), we have that $\hat{\boldsymbol{w}}, \hat{y} \in L_l$.

Choose any leaf l, such that $\exists(\boldsymbol{x}, \boldsymbol{w}, y) \in \mathcal{L}_l \cap \mathcal{A}$. Take any $\hat{\boldsymbol{z}} = \boldsymbol{e}_l$, which results from constraints (4.19d), (4.19e). Then $\hat{\boldsymbol{x}} \in \mathcal{A}$ from constraint (4.9b), and $\hat{y} = s_l$. For each dimension $i \in \{1, .., d\}$:

- If $l \in H'_i$ then (4.21a) implies $1 \leq w_i$, while (4.21b) implies $1 \geq w_i$.
- If $l \in T'_i \setminus H'_i$ then (4.21a) implies $0 \leq w_i$, while (4.21b) implies $1 \geq w_i$.
- If $l \notin T'_i$ then (4.21a) implies $0 \leq w_i$, while (4.21b) implies $0 \geq w_i$.

Therefore $\hat{\boldsymbol{w}}, \hat{y} \in L_l$. □

Although this extension may seem fairly straightforward, it actually removes a significant number of fractional solutions from the linear relaxation of (4.9) compared

to using formulation (4.19) to represent the tree. Even though the formulation is not ideal, it has many favorable properties relative to the formulation in chapter 3. It has fewer constraints and variables, which generally result in faster MIP solve times. Furthermore it has no big-M constraints, which likewise can lead to slow solve times if M is not chosen carefully. Finally the fact that there is a subset which is ideal suggests it is a 'tighter' formulation. Section 4.7.1 shows the performance of the two methods in more detail with numerical simulations.

4.6.4 Extensions to tree ensembles

This formulation can be extended to tree ensembles, such as random forests. If there are $q \in \{1, ..., Q\}$ trees, then the notation can be extended to z_{lq} being a binary variable indicating if leaf l in tree q is active and $\mathcal{T}_q^{\text{binary}}$ as the MIP formulation describing the tree. Then the formulation becomes:

$$\max_{\boldsymbol{x}} \sum_{i=1}^{D}\sum_{j=1}^{D} r_{ij}x_{ij} + \sum_{q=1}^{Q} y_q \tag{4.22a}$$

$$\text{s.t.} \quad \boldsymbol{x} \in \mathcal{A} \tag{4.22b}$$

$$\boldsymbol{w}, y_q \in \mathcal{T}_q^{\text{binary}} \qquad \forall q \in \{1, ..., Q\} \tag{4.22c}$$

$$w_i = l_{it}^N - \sum_{j=1}^{D} x_{ij} + \sum_{j=1}^{D} x_{ji} \qquad \forall i \in \{1, ..., D\} \tag{4.22d}$$

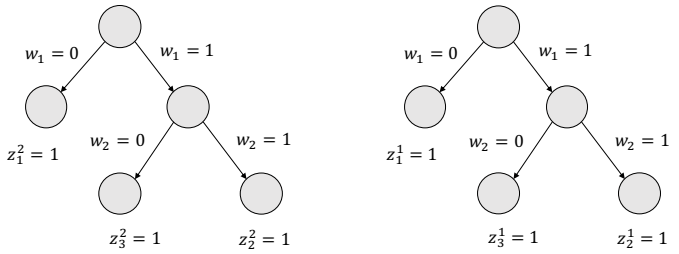
There is another negative result (Example 2) which shows that although $\mathcal{T}_q^{\text{binary}}$ is ideal, $\cap_{q=1}^{Q}\mathcal{T}_q^{\text{binary}}$ is generally not ideal.

Example 2 ($\mathcal{T}_1^{\text{binary}} \cap \mathcal{T}_2^{\text{binary}}$ is not ideal)**.** *Suppose we have the following formulations for two trees in an ensemble:*

$$z_2^1 + z_3^1 \leq w_1 \tag{4.23a}$$

$$z_2^1 + z_3^1 \geq w_1 \tag{4.23b}$$

$$z_2^1 \leq w_2 \tag{4.23c}$$



(a) Tree 1 (b) Tree 2

Figure 4-2: Example: tree ensemble formulation is not ideal

$$z_1^1 + z_2^1 \geq w_2 \tag{4.23d}$$

$$z_2^2 + z_3^2 \leq w_1 \tag{4.23e}$$

$$z_2^2 + z_3^2 \geq w_1 \tag{4.23f}$$

$$z_2^2 \leq w_2 \tag{4.23g}$$

$$z_1^2 + z_2^2 \geq w_2 \tag{4.23h}$$

$$z_1^1 + z_2^1 + z_3^1 = 1 \tag{4.23i}$$

$$z_1^2 + z_2^2 + z_3^2 = 1 \tag{4.23j}$$

$$w_1,\ w_2,\ z_1^1,\ z_2^1,\ z_3^1,\ z_1^2,\ z_2^2,\ z_3^2 \geq\ 0 \tag{4.23k}$$

$w_1 = 0.5, w_2 = 0.5, z_1^1 = 0.5, z_2^1 = 0.5, z_3^1 = 0, z_1^2 = 0.5, z_2^2 = 0, z_3^2 = 0.5$ *is a basic feasible solution, which does not have binary variables.*

However, this formulation does offer some advantages over the formulation from chapter 3. The advantages are similar to those mentioned previously in Section 4.6.3. We explore this in more detail with numerical experiments in Section 4.7.1.

4.6.5 Enumerate then optimize

Another method for optimizing trees subject to assignment constraints is to use an enumerative approach. This is an approach we also benchmark against in Section 4.7. This proceeds by solving individual assignment problems for each leaf as follows:

$$\max_{l=1...p} \max_{\boldsymbol{x}} \sum_{i=1}^{D} \sum_{j=1}^{D} r_{ij} x_{ij} + y_l \tag{4.24a}$$

$$\text{s.t. } \boldsymbol{x} \in \mathcal{A} \tag{4.24b}$$

$$\boldsymbol{w}, y \in \mathcal{W}_l \tag{4.24c}$$

$$w_i = l_{it}^N - \sum_{j=1}^{D} x_{ij} + \sum_{j=1}^{D} x_{ji} \qquad \forall i \in \{1, ..., D\} \tag{4.24d}$$

Therefore, for a tree with p leaves, p assignment will have to be solved, and the leaf with the maximum optimal solution selected. Many of the leaves will be infeasible and can be pruned. This approach has the advantage that the subproblem that needs to be solved for each leaf is an LP so can be solved quickly, but there are very many such leaves so it can still take a significant amount of time to solve, as explored in Section 4.7.1.

4.7 Numerical experiments

In this Section, we test both the performance of different approximate dynamic programming algorithms in terms of finding good (but not necessary optimal) policies. We also test the formulations from Section 4.6 to find how fast they can solve the subproblems from the approximate DP. Being able to solve the intermediate subproblems extremely fast is crucial in a dynamic programming setting which may require several rounds of policy or value iteration over a large state space. We implemented the algorithms in the `Python` programming language (Rossum (1995)), using a `Gurobi` solver (Gurobi Optimization (2019)). The experiments were run on a macbook with an Intel Core i5@2.4GHz with 8GB RAM, and all tests were limited to a single thread.

4.7.1 Solution time

In this section, we test the speed to solve optimization problems related to nurse practitioner routing with approximation of the cost to go with trees/tree ensembles to

optimality using different formulations and algorithms. This is important considering that these subproblems are solved many times in an iterative dynamic programming framework.

Solution time for single tree and assignment polytope:

We begin with modeling the cost to go function with a single tree. We test the time to solve to optimality for a number of different randomly generated problem instances. We generate a random network of demand nodes uniformly within the unit square, where the number of demand nodes is twice the number of nurse practitioners. Patients are randomly generated and assigned uniformly (without replacement) to a subset of the demand nodes. The number of patients to be served in the next period is half the number of nurse practitioners. A CART regression tree is trained to represent the reward associated with the future location of the nurse practitioners. The data used to train the tree is also randomly generated, with each location vector having a patient in a location with probability 0.25 and no patient with probability 0.25, and a uniform random reward between 0 and the number of customers. The size of the dataset used to generate the tree is $n = 200000$. As the focus is on investigating the tightness of the formulations, `Gurobi` presolve routines are turned off. We compare `enumerate_and_opt` (formulation 4.27), `big_M` (formulation (P) form chapter 3), and `union_of_polyhedra` (formulation 4.19). We test over a grid of a different number of nurse practitioners $\{10, 100, 500\}$ and tree depth $\{5, 10, 13, 16\}$, which results in trees with $\{63, 2047, 17605, 57367\}$ nodes respectively. The time taken is the time to solve to optimality to find the best solution.

We observe in figure 4-3 that in all cases the `union_of_polyhedra` outperforms the other methods, often by an order of magnitude or greater (note the non-linear scale on the y-axis). This is more noticeable on smaller problem instances. The `big_M` method tends to perform well relative to `enumerate_and_opt` for instances with a small tree but a large assignment problem, but performs poorly when the size of the tree is large. Note, `enumerate_and_opt` and `big_M` reached the time limit of 100s for 500 nurse practitioners and depth 16, while `enumerate_and_opt` also hit

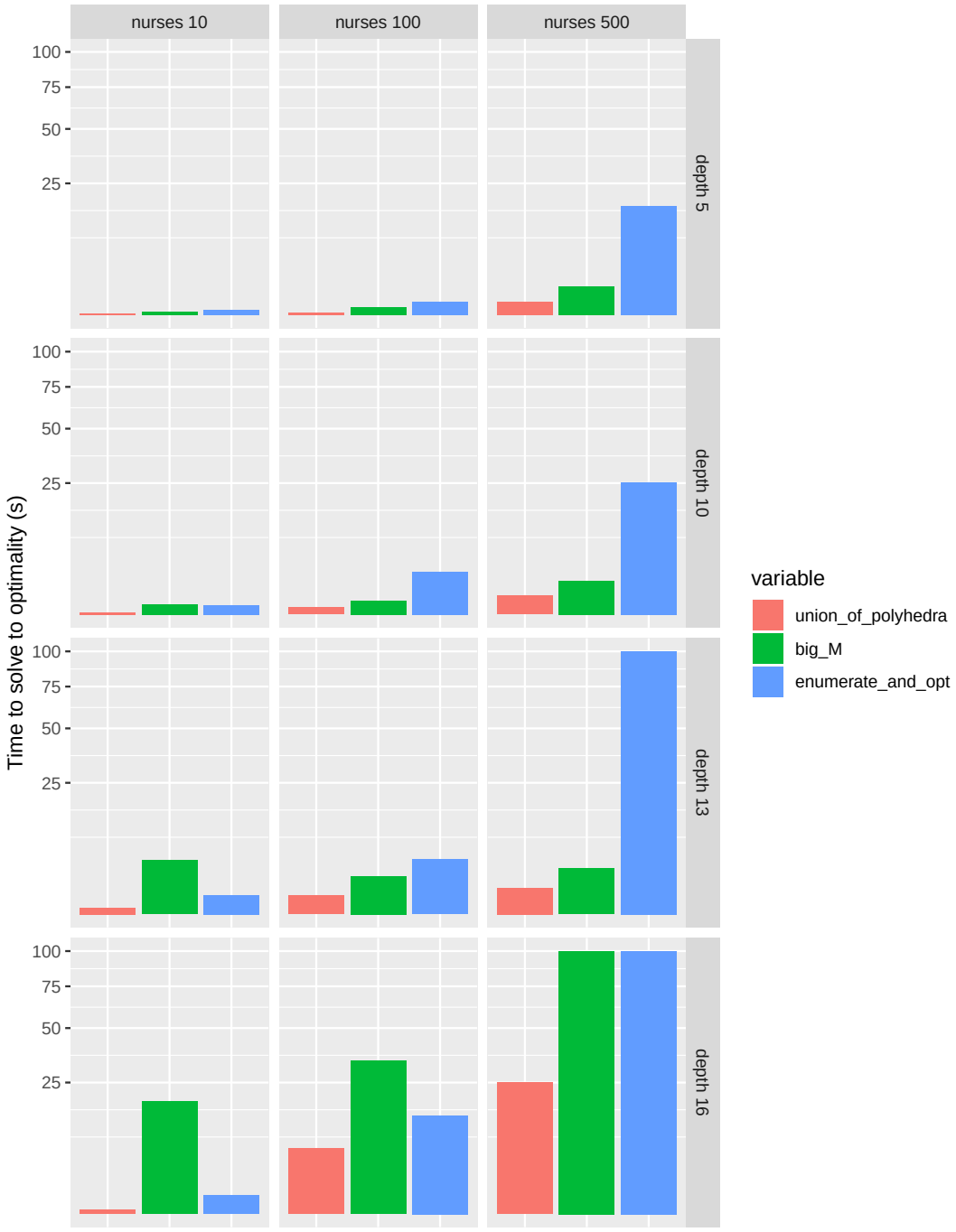
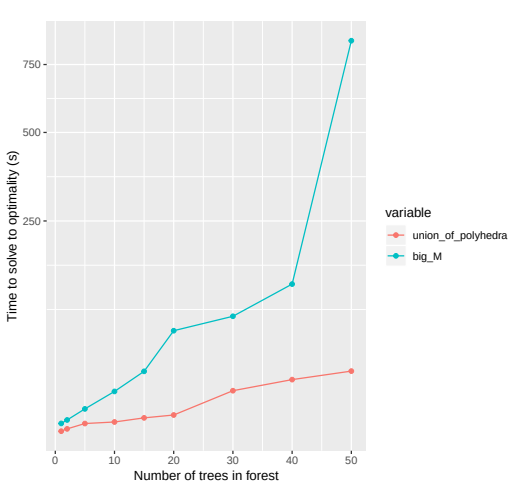
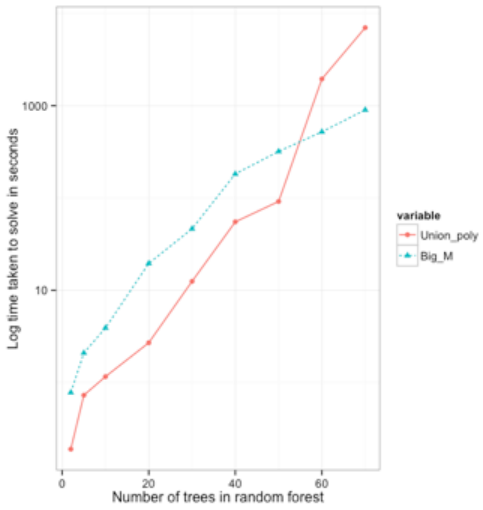


Figure 4-3: Time taken to solve for single tree with varying size of assignment and tree size



(a) forest with assignment constraints



(b) unconstrained forest

Figure 4-4: Time taken to solve forest

the time limit for 500 nurse practitioners and depth 13.

Solution time for forest and assignment polytope

Next we show numerical experiments to explore the time to solve to optimality when the cost-to-go function is approximated by a random forest. Figure 4-4b follows the experimental set-up of Section 4.7.1, except a random forest is trained from the randomly generated data rather than a CART tree. The depth of the trees in the forest is capped at 10, and there are 100 nurse practitioners to be routed. The number of trees in the forest is increased from 2 to 50. We observe that in this constrained example, the `union_of_polyhedra` is significantly faster than the `big_M` formulation over instances we experimented on.

For completeness, we also note that the `union_of_polyhedra` formulation is not always more suitable that the `big_M` formulation for use in forests. Figure 4-4b, shows the performance of the two methods on the wine dataset from the UCL machine learning repository (Dua and Graff (2017)). We maximize quality with no constraints on wine produced. We observe that for small forests, the union of polyhedra approach is faster, but for large random forests the `big_M` formulation is faster. This is possibly because the `big_M` formulation has advantages in the branching of

the integer programming solver. The `big_M` formulation has variables $\boldsymbol{q}$ associated with each node/branch in the tree, whereas the `union_of_polyhedra` formulation only has variables associated with each leaf $\boldsymbol{z}$. Branching on the node/branch variables $\boldsymbol{q}$ in that formulation is powerful, as setting $q_i = 0$ as it sets many leaf variables z_j which are children of that branch to be zero. In forests with few trees, most of the time is spent on presolve and cutting plane generation where the tighter union of polyhedra has an advantage, while in forests with a large number of trees, a greater time is spent on branching where the big-M method has an advantage. Another benefit is that Benders decomposition can be applied to the `big_M` formulation which offers additional speed up, while it cannot be applied to the `union_of_polyhedra` formulation.

4.7.2 Performance of approximate dynamic programming approaches

In this section we test the methods from Section 4.5 in their ability to find good policies for the nurse practitioner routing problem. We test two objectives of interest; maximizing the number of patients served in a high demand setting and minimizing the distance that the nurse practitioners travel in a lower demand setting.

We construct a simulation environment to test the performance of the policies. The area being serviced is a square with a 60 minute travel between opposite edges. The demand is discretized into a number of uniformly distributed demand nodes equal to twice the number of nurse practitioners who are in service. Demand at each node follows a Poisson distribution, with an overall arrival rate of twice the number of nurse practitioners per hour for low traffic setting and four times the number of nurse practitioners per hour for heavy traffic. We set a fixed 20 minute service time and a 60 minute maximum wait time for each patient.

As mentioned in Section 4.5, we do one step of direct policy iteration, and also extend to another step for maximizing the distance travelled. To do this, patients are randomly generated and served using the myopic LP policy (4.8), denoted by `reopt`.

Every time a patient request arrives an assignment of nurse practitioners is made, the location of the nurse practitioners is recorded as a feature vector x_i, using a basis function approximation of the full state space. The number of patients served within the next 60 minutes and the distance that the nurse practitioners' travel is recorded as a response y_i. We then train a function to predict y as a function of x, using CART and linear regression packages in `scikit-learn` (Pedregosa et al. (2011b)). These functions are used as the cost-to-go function approximation when evaluating the decision of how to serve future patients in formulation 4.5. These policies are denoted by `reopt+linear` and `reopt+linear`. In addition, we benchmark against the myopic LP policy (4.8). We then simulate the performance of each method over additional patients and record the number of patients seen and the distance travelled of the nurse practitioners.

Minimizing the distance travelled

Figure 4-5 shows the average distance the nurse practitioners travel to meet demand. This is in a lower demand setting where the number of nurse practitioners is sufficient to serve all patients, so the system operator is able to focus on secondary objectives. In this formulation, $r_{ij} = \theta - d_{ij}$, with the θ term being sufficiently large such that all patients will be served where possible, and the future distance travelled is a secondary consideration.

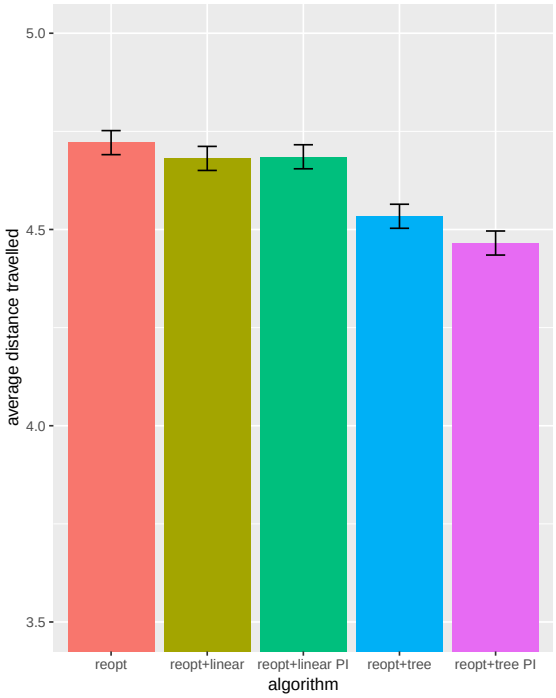
In this section we also test a policy iteration approach, where in addition to estimating value functions based on data from the reoptimization policy, we also estimate the value functions after a round of following the policy of `reopt+linear` and `reopt+tree` respectively. This produces policies `reopt+linear PI` and `reopt+linear PI`. A discount factor of 0.5 is used in all simulations. The number of nurse practitioners used varies from $\{4, 8, 15\}$, which is in-line with how many nurse practitioners our collaborator uses at any given time. The number of customers used per training round is $\{10000, 50000, 100000\}$ respectively and the same number is used for testing.

Using a cost-to-go function which is based on a tree results in policies which are

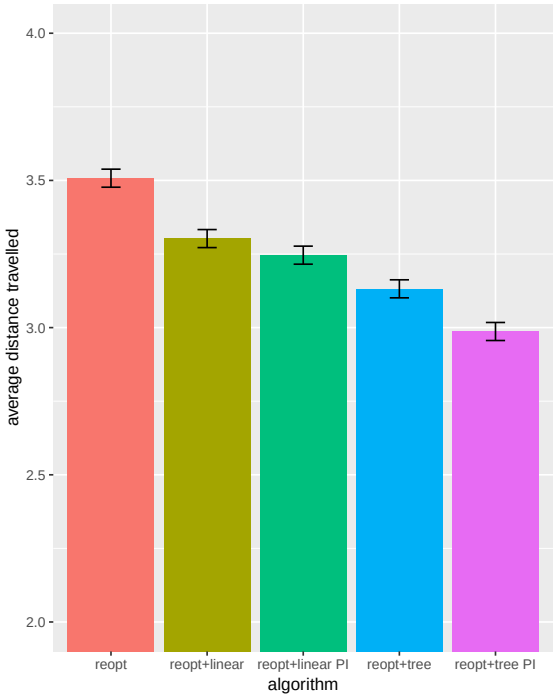
better than the myopic policy or the policy based on a linear approximation of the cost-to-go function, which holds over different problem sizes. The policy iteration step also corresponds to improved outcomes relative to using data generated from the myopic policy. Part of the reason the tree based approximation methods outperform the linear regression based approximation is that the cost-to-function is inherently non-linear in this setting. The linear function does not capture the interaction effects between nurse practitioners in adjacent locations, who are able to serve patients that arise at nearby demand nodes. Another advantage of using trees to approximate the cost-to-go function is that through the training and pruning procedure a tree will only capture the interaction effects which are significant. In all generality, a tree is able to recreate a look-up-table as is used in traditional dynamic programming, but through appropriate regularization, only the interactions which are important are stored. This can be thought of as automated basis function selection in a way. This greatly simplifies and speeds up solving the problem.

Maximizing the number of patients served

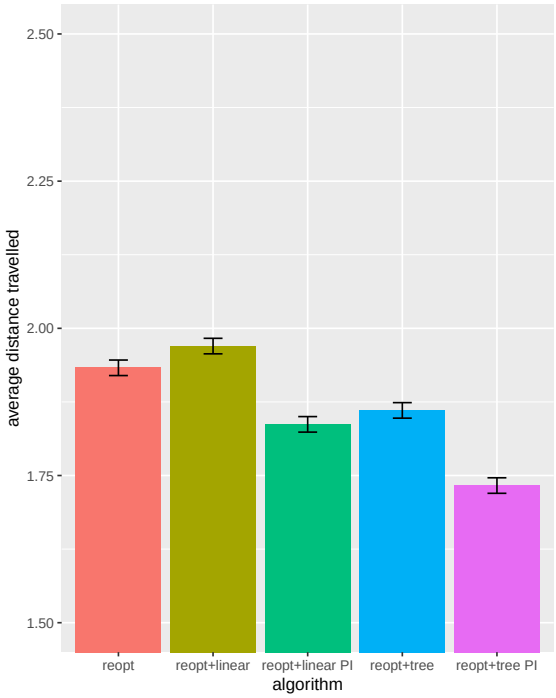
In figure 4-6 we observe the number of patients that each algorithm is able to serve for problems of varying size. The simulations take place in a high demand setting where the number of nurse practitioners is not adequate to serve all patients, so decisions must be made about which patients to serve and which to reject. In this formulation, $r_{ij} = 1$, so the priority is serving as many patients as possible. In these simulations, we use as a basis function the time and location at which the nurses practitioners will become available, which was more effective in this setting. This approach is outlined in section 4.5. In these simulations we also compare against a value function estimated using a random forest `reopt+RF`. As the random forest is significantly slower to optimize over, doing a policy iteration approach is not feasible. The number of nurse practitioners used varies from $\{4, 8, 15\}$, with training sets increasing in size $\{10000, 50000, 100000\}$. The testing set was 500 patients, smaller than in figure 4-5, again due to the random forrest speed. As figure 4-6 shows, using a cost-to-go function which is based on a tree or forest results in policies which are better than



(a) 4 Nurse Practitioners



(b) 8 Nurse Practitioners



(c) 15 Nurse Practitioners

Figure 4-5: Minimizing distance travelled

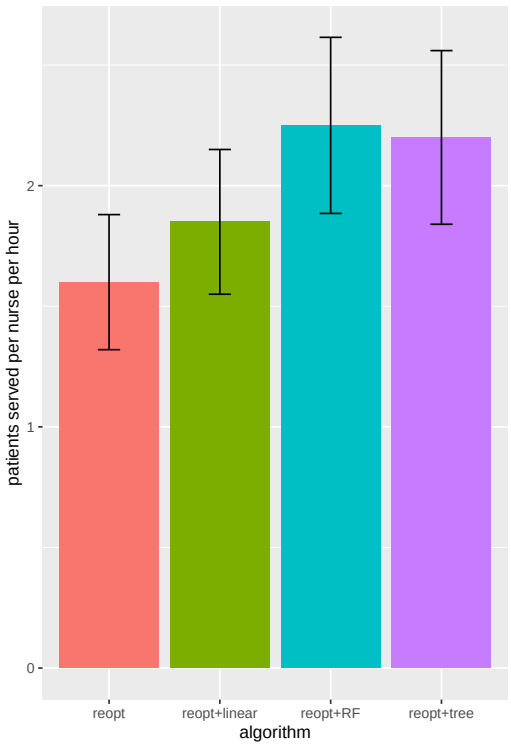
the myopic policy or the policy based on a linear approximation of the cost-to-go function. This holds across all problem sizes. The tree based approach appears to perform comparably to the random forest approach suggesting that there is little advantage to using an ensemble in this setting.

4.7.3 Insights

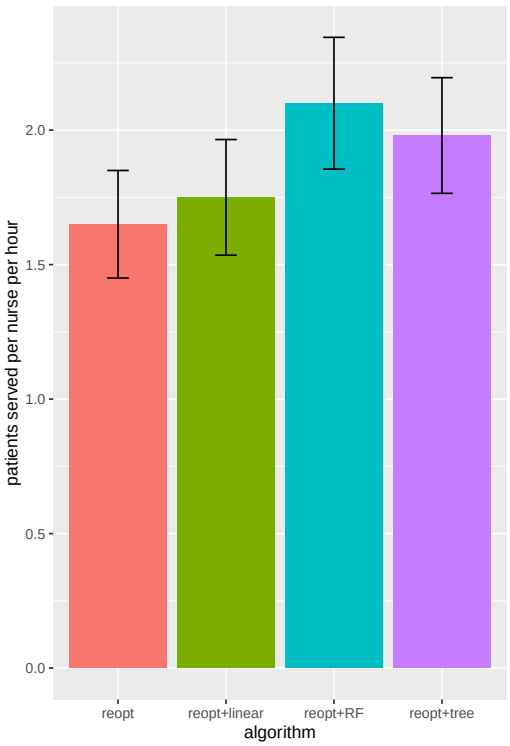
We can further investigate the how the system behaves when nurse practitioners are routed by focusing on the a tree approximation for the cost-to-go function, and in particular what the distributions of patient wait and nurse travel time are. In the following simulations, the simulation environment is similar to the previous section, except that the rate of service is held constant while the number of nurses varies. In particular, the arrival rate of patients is 12 per hour, over a discretization of 20 demand nodes. 10000 patients are used to train the trees, while 1000 are used for testing. The objective function used is minimizing distance travelled by the nurse practitioners.

Figure 4-7 shows the patient wait times, nurse travel times and number of patients served as the number of nurse practitioners increases. The number of patients able to be served increases when the number of nurse practitioners increases from 2 to 9, at which point all patients are served. The patient wait times and distance the nurses travel per patient is relatively constant from $2 - 6$ nurse practitioners but falls with each additional nurse practitioner introduced after that. Counterintuitively, The distance travelled and wait times are not strictly decreasing functions with number of nurse practitioners used because in very high demand situations the algorithm will prioritize serving patients who are easier to serve and rejecting those that aren't. These are typically patients which are close to the current locations of nurse practitioners, therefore requiring less travel and lower wait times.

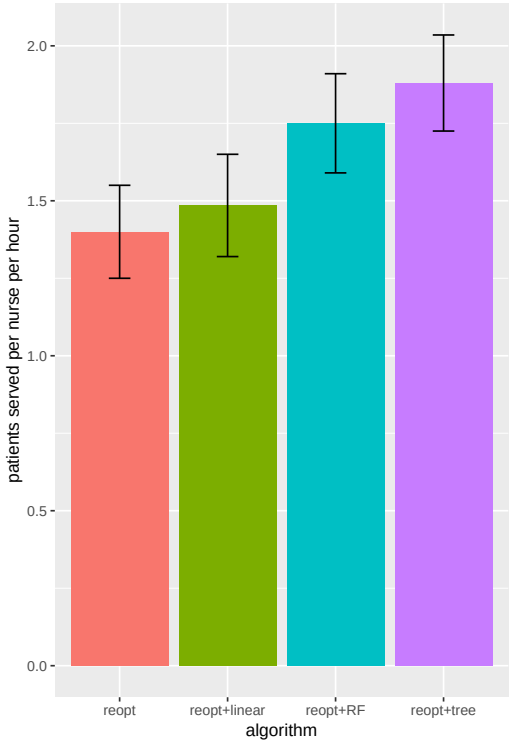
These simulations are useful for our collaborators for planning purposes. Different levels of staffing can be tested to see how metrics of interest are impacted under different demand scenarios. This can lead to the selection of the correct number of nurses to schedule for any given period during the day, and reduce the uncertainty



(a) 4 Nurse Practitioners



(b) 8 Nurse Practitioners



(c) 15 Nurse Practitioners

Figure 4-6: Maximizing number of patients served

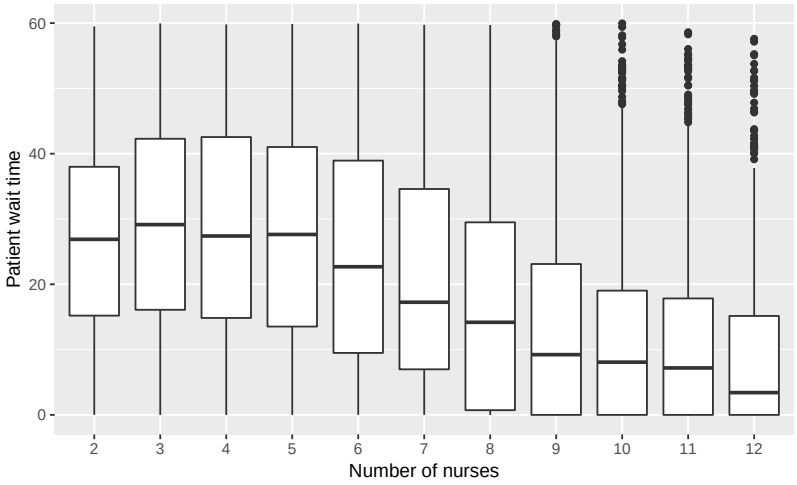
they currently face when making such decisions.

4.8 Extensions to other operations management problems

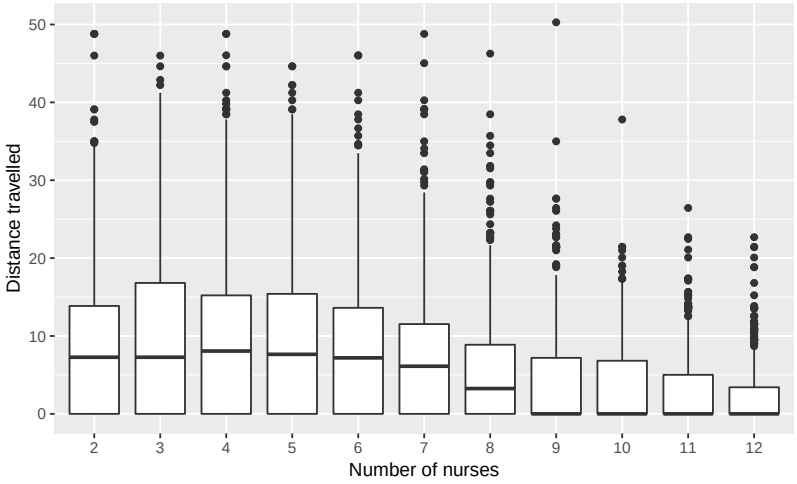
In addition to nurse practitioner routing, there are many other problems to which the techniques from this chapter are applicable. These problems share two characteristics; i) they are online, so multiple decisions need to be made as information is revealed and ii) each decision has a complex combinatorial structure. In this section we look at a general online matching problem. We first look at a classic problem of matching pairs of vertices on a general undirected graph. We then extend to the case of matching up to three vertices together to form non-overlapping groups. This formulation can be applied to online decision making problems such as kidney exchange or car pooling.

In the case of kidney failure, even if the patient in need of a kidney is able to find a donor, often that donor will not be compatible with the patient. Kidney exchanges address this problem. In an exchange, a donor/patient pair will be matched with another donor/patient pair, such that the donor in the first pair is compatible with the patient in the second pair, and vice versa. As a result of this process, both patients end up with kidneys, but not from their original donor. This can also be extended to from a *3—cycle*, which involves 3 donor/patient pairs forming a cycle of compatible kidneys exchanges. This process greatly expands the number of successful kidney transplants that take place. For a complete review of this process, see Sönmez and Ünver (2017). In the context of the following problem, we consider each donor/patient pair being a vertex, which will be matched to another to form a 2—cycle, or to form a group of 3 in a 3-cycle. The reward for a matching may correspond to how compatible each patient is with their corresponding donor, and the risk of the transplant is not successful in the long run.

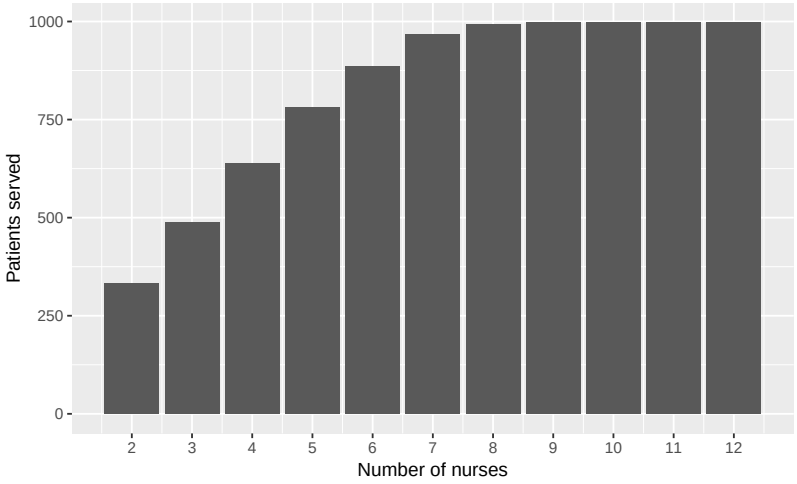
This framework can also be applied to carpooling, such as the services currently offered by ride-sharing apps *Lyft* and *Uber*. In this instance we can think of each



(a) Patient wait times



(b) Nurse distance travelled



(c) Number of patients served

Figure 4-7: Distribution of wait and distance using tree approximation

vertex as being a customer with an origin (pick-up) and destination (drop-off). If a pair of riders is matched together, the driver will pick up the both customers before dropping off either customer. The reward would correspond to some function of total distance travelled and may possibly include waiting time and fairness considerations. This reward may need to be computed as a pre-optimization step — for instance the shortest distance travelled to serve both customers may need to be calculated in advance. This problem can also be extended to matching three customers together in a car, where they would all share the ride. Variations also exist where the driver is counted as a vertex.

4.8.1 2-way online matching problem

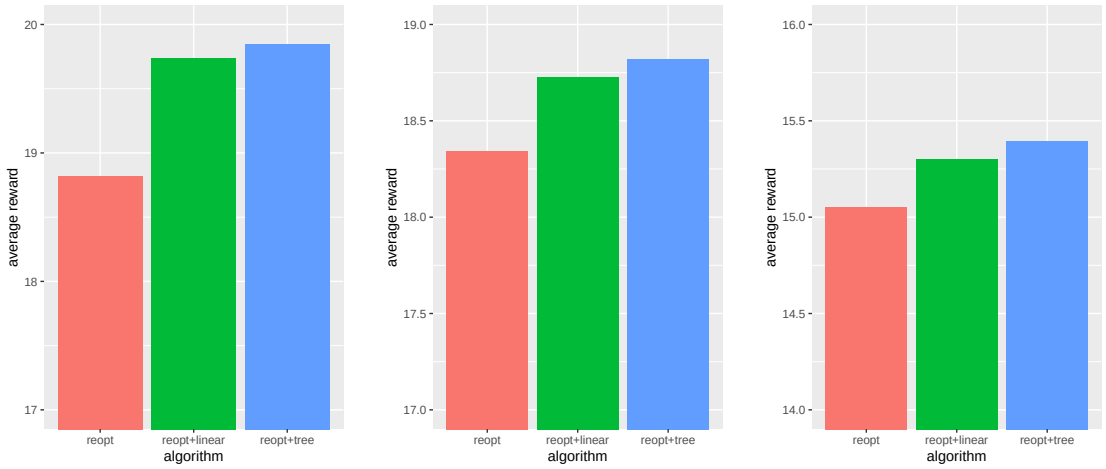
We begin by looking at the simpler formulation of matching pairs together. Suppose we have n types of item. At each stage there is an availability vector a_i which is equal to the number of items of type i available to be matched. Define $x_{ij} = 1$ if we match item i to j, which exists for $i < j,\ i, j \in \{1, ..., p\}$, for which there is a reward r_{ij}. We can solve an optimization formulation to find the optimal matching:

$$J(a) = \max_{\boldsymbol{x}} \sum_{i=1}^{n} \sum_{j=1}^{i} r_{ij} x_{ij} \tag{4.25a}$$

$$\text{s.t.} \quad \sum_{j=1}^{i} x_{ij} + \sum_{j=i}^{n} x_{ji} \leq a_i \qquad \forall i \in \{1, .., n\} \tag{4.25b}$$

$$x_{ij} \in \mathbb{Z}_+ \tag{4.25c}$$

In this formulation, constraint (4.25b) enforces that each item i is only used once. We solve a two stage matching problem. There is an first matching period, then out of the unmatched items, each will depart with probability μ. There will also be new arrivals of each type with probability λ. A second matching problem is then solved to match the available items. At the time of the first decision, these departures and arrivals are not known, and the unmatched items at the end of the first period are w_i,



(a) 4 types (b) 6 types (c) 8 types

Figure 4-8: 2 way online matching problem

enforced by constraint (4.26c). The formulation for the two stage matching problem is:

$$\max_{\boldsymbol{x}} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} r_{ij} x_{ij} + E[J(w)] \tag{4.26a}$$

$$\text{s.t.} \quad \sum_{j=i+1}^{n} x_{ij} + \sum_{j=1}^{i-1} x_{ji} \leq a_i \qquad \forall i \in \{1, .., n\} \tag{4.26b}$$

$$a_i - \sum_{j=i+1}^{n} x_{ij} + \sum_{j=1}^{i-1} x_{ji} = w_i \qquad \forall i \in \{1, .., n\} \tag{4.26c}$$

$$x_{ij} \in \mathbb{Z}_+ \tag{4.26d}$$

The expectation encompasses the transition process as described. We approximate $E[J(w)]$ using a linear function, a tree function and compare to a myopic approach where $E[J(w)] = 0$. We run simulations to compare the policies formed by these approximations. In the simulations $\lambda = 0.15$, $\mu = 0.1$ and r_{ij} is a randomly generated matrix with an exponential distribution with mean 2. A training set of 40000 samples were randomly generated to train $E[J(w)]$. A sample of 3000 trials were used to evaluate the methods.

Figure 4-8 shows the performance of the approaches on in the matching setting

for instances with 4, 6 and 8 types. Using a tree based cost-to-go approximation outperforms the linear approximation, which in turn does better than the myopic approach.

4.8.2 3-way online matching problem

We can adjust formulation (4.26) to allow 3-way matchings in addition to two way matchings. In this formulation x_{ijk} refers to whether item i, j and k are matched together as a triplet for $i < j < k,\ i, j, k \in \{1, ..., p\}$, while x_{ij} still refers to item i and j being matched together as a pair for $i < j,\ i, j \in \{1, ..., p\}$.

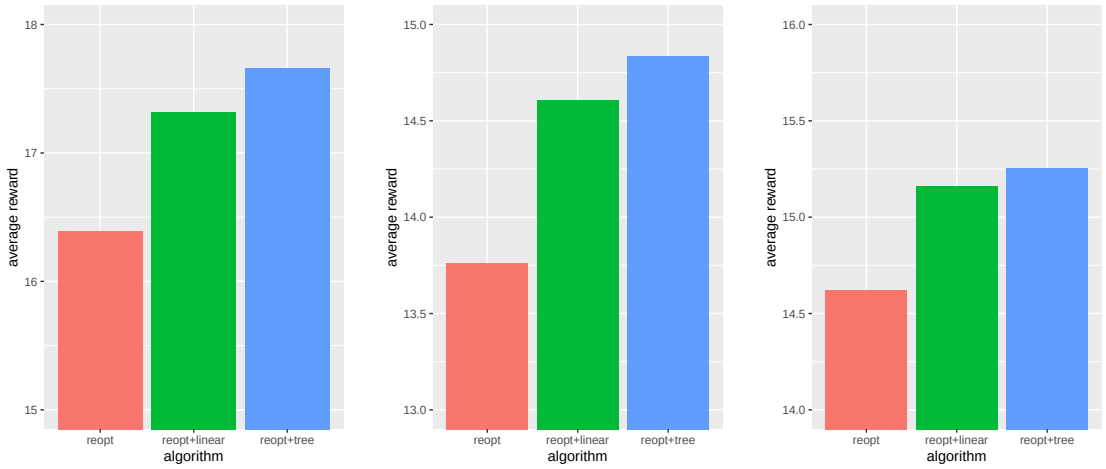
$$\max_{\boldsymbol{x}} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n-1} \sum_{k=j+1}^{n} r_{ijk}^{(3)} x_{ijk} + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} r_{ij}^{(2)} x_{ij} + E[J(w)] \tag{4.27a}$$

$$\text{s.t.} \sum_{k=i+2}^{n} \sum_{j=i+1}^{n-1} x_{ijk} + \sum_{k=i+1}^{n} \sum_{j=1}^{i-1} x_{jik} + \sum_{j=1}^{i-2} \sum_{k=j+1}^{i-1} x_{jki} + \sum_{j=i+1}^{n} x_{ij} + \sum_{j=1}^{i-1} x_{ji} \leq a_i \qquad \forall i \in \{1, .., n\} \tag{4.27b}$$

$$a_i - \sum_{k=i+2}^{n} \sum_{j=i+1}^{n-1} x_{ijk} + \sum_{k=i+1}^{n} \sum_{j=1}^{i-1} x_{jik} + \sum_{j=1}^{i-2} \sum_{k=j+1}^{i-1} x_{jki} + \sum_{j=i+1}^{n} x_{ij} + \sum_{j=1}^{i-1} x_{ji} = w_i \qquad \forall i \in \{1, .., n\} \tag{4.27c}$$

$$x_{ij}, x_{ijk} \in \mathbb{Z}_+ \tag{4.27d}$$

Similar to the two-way matching formulation, constraint (4.27b) and (4.27c) restrict each item to being matched once and toestablish how many items will be available in the second stage respectively. The simulation environment used was also similar with $\lambda = 0.15, \mu = 0.1$, $r^{(2)}$ and $r^{(3)}$ randomly generated matrix with an exponential distribution with mean 2 and 4 respectively to encode that three way matchings are more valuable than two way matchings. The training set had 40000 samples, while the evaluation set had 3000 trials.



(a) 4 types (b) 6 types (c) 8 types

Figure 4-9: 3 way matching problem

Figure 4-9 shows the performance of the approximation techniques for the 3 way matching problem. Similar to the 2 way matching approaches, the tree based approximation does better than the other approximations, but the difference between the linear and the tree approximation becomes smaller as the size of the matching increases. Also of note is that there appears to be a greater difference between the linear and tree based approximation in the 3 way matching relative to 2 way, possibly because the second stage is inherently more non-linear with combinations of 3.

4.9 Conclusions

In this chapter, we address the common problem of solving approximate dynamic programs with non-linear cost-to-go functions when the action space is too large to enumerate, but has structure which can be modeled using mixed integer optimization. We show that using a tree approximation can lead to formulations which are not only accurate in predicting future cost, but are also tractable to solve. To achieve this, we have introduced new mixed integer optimization formulations for optimizing over trees/tree ensembles and have shown favorable properties of these formulations, both theoretically and experimentally. We show these results using a real-world nurse practitioner routing problem as a case study and show that we are able to outperform

other approximations from the approximate dynamic programming literature.

Chapter 5

A ranking algorithm for tramp shipping in the spot market

5.1 Introduction

The shipping industry is the backbone of the global economy, with 90% of international trade transported by sea. The United Nations Conference on Trade and Development estimated that the operation of merchant ships contributes about US $380 billion in freight rates within the global economy, equivalent to about 5% of total world trade. With each cargo ship costing between $20-200 million, there is significant interest in ensuring these valuable resources are employed in the most effective way possible.

This chapter is motivated through a collaboration with a large tramp shipping company. Tramp shipping is analogous to a taxi service whereby customers who want cargoes (typically oil) transported form an agreement with a shipping company to do so. The literature splits cargoes into two classes; contract and spot (Christiansen et al. (2013)). Contract cargoes are negotiated far in advance and the shipping company has an obligation to pick them up, while spot cargoes are put on the market on short notice. Most of the models in the tramp shipping literature focus on contract cargoes (see Brønmo et al. (2007), Malliappi et al. (2011), Korsvik et al. (2010), Brønmo et al. (2007)). However, for many shipping routes, observed lead times (time between

signing a contract and picking up the cargo) are short, so shipping companies do not have the ability to plan far in advance. For example, on the route for crude oil between the Middle East and Japan, analysis of historical voyages show that the average lead time is 11 days. This is roughly equivalent to the time taken to sail between the Middle East and Japan, confirming that in some markets cargoes agreed to at short notice are the norm. There is a limited number of papers which address the stochastic and dynamic nature of spot cargoes (see Tirado et al. (2013), Hwang et al. (2008), Cheng and Duran (2003)). Unfortunately, the time it takes to solve the complex models introduced in these papers is impractical. In addition, it is difficult to tell how the heuristics proposed in the literature compare to the optimal solution. These limitations are discussed in detail in section 5.1.2. In comparison, our goal in this chapter is to propose a simple algorithm which has fast performance and for which we establish an optimality guarantee on its performance.

To further complicate the decision, the payment received for transporting a cargo can vary according to fluctuations in the market. Figure 5-1 shows a snapshot of the time charter equivalent rates for a particular trading route, which are a proxy for daily revenue. As can be observed, the shipping rates are highly variable. This differentiates tramp shipping from several other modes of transportation, such as long distance trucking where prices are assumed to be known in advance. This is a very competitive market with many players. For example, in the VLCC (very large crude carrier) fleet, there are 638 ships and the largest player has 40 ships. This chapter aims to devise a method to make effective routing decisions when choosing between spot cargoes with minimal foresight into future cargoes.

5.1.1 Problem description

We study the problem of deciding which spot cargo a shipping broker should choose, assuming she has not committed to any contract cargoes. As a ship discharges at a port, brokers search for which spot cargo to pick up next. This is consistent with the short lead times observed in practice. Upon discharge, the broker observes an independent random set of cargoes, and chooses a cargo from the set of available

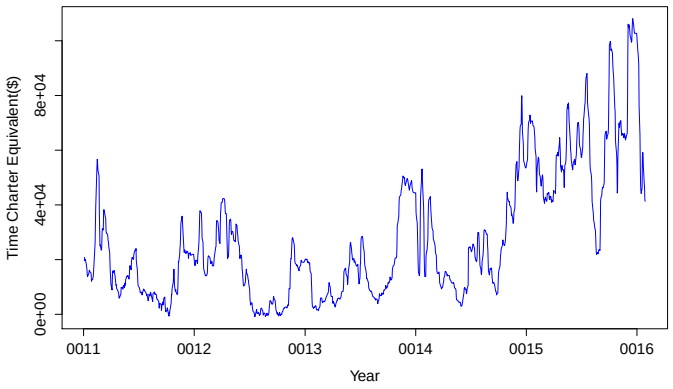


Figure 5-1: Shipping rates for the TD3 route (between Middle East and Japan) illustrating the variability in shipping rates

cargoes. This poses a challenge because there is a potentially large number of different cargo combinations a broker has to choose from. Once the broker commits to a cargo, the profit from that cargo is received, and a period of time equivalent to the length of the voyage will elapse before the broker will make the next decision. Our objective is to maximize total profit over a finite time horizon. We model this scenario with a single ship, as is done in practice. In fact, we saw little overlap between cargoes offered to ships under the same management due to the fact that the market is made up of many relatively small players with few ships who are unlikely to be loading in the same area at the same time. We first model the scenario where shipping rates over this period are assumed to follow a known forecast, so that only the cargo availability is uncertain and not shipping rates. We then extend this model to include uncertainty in shipping rates in Section 5.7.

5.1.2 Relevant Literature

Deterministic tramp shipping

The models in the literature are typically deterministic, where all cargoes are known ahead of time. The goal is to design routes for each ship such that all contract cargoes are picked up within a given time window, and (known) spot cargoes may be incorporated into the schedule if it is profitable to do so. A popular formulation was

presented by Brønmo et al. (2007), as a Mixed Integer Linear Program (MILP) which shares similarities with the more general multiple vehicle routing formulation from Lau et al. (2003). As this MILP is difficult to solve for large problem sizes, several heuristics have been proposed to obtain approximate solutions (see Malliappi et al. (2011), Korsvik et al. (2010), Brønmo et al. (2007)). These models may perform well for companies who operate with mainly contract cargoes, but our experience suggests that the tramp shipping market is much more dynamic in practice, with shipping brokers (who plan and manage ship schedules) having to constantly adjust their schedules as spot cargoes become available and market conditions change. As our focus in this chapter is on spot cargoes, our approach is dynamic and stochastic.

Dynamic and stochastic tramp shipping

There have been a limited number of attempts to model uncertainty in the tramp shipping market. Tirado et al. (2013) present a tramp shipping adaptation of the model by Bent and Van Hentenryck (2004b), who apply their algorithms to a dynamic vehicle routing problem. These algorithms involve sampling sets of spot cargoes over the time horizon, and solving the MILP from Brønmo et al. (2007) heuristically for each sample set of cargoes. Brønmo et al. (2007) then look at the different routes taken in the first period, and score each route according to a function of the number of times it was taken over the samples, and the value of those routes. They then choose the first period route with the highest score. Due to the complexity of the problem, Brønmo et al. (2007) use a heuristic approach. As a result, there is no guarantee on convergence to the optimal solution. In comparison, we study a problem that focuses on a shipping company whose core business is spot cargoes. Furthermore, for the setting we study, we are able to solve this problem to optimality.

Hwang et al. (2008) solve a problem where there is uncertainty in the future shipping rates but no uncertainty in cargo availability. Hwang et al. (2008) proposes a set packing model that limits risk using a quadratic variance constraint, and solves the problem using a branch-price-and-cut method. In contrast we model a scenario where cargo availability is uncertain ahead of time. Cheng and Duran (2003) formulate an

ambitious Markov Decision Process for ship routing including fleet size and composition decisions. They use a simulation and dynamic programming approach but are unable to solve for realistic problem sizes. For a full review of maritime routing and fleet management problems, we refer the interested reader to the review paper by Christiansen et al. (2013) and references therein.

Other dynamic transportation

The tramp shipping problem we consider also shares similarities with other transportation problems. There are many examples of dynamic programming being applied to dynamic resource allocation and fleet management (e.g. Godfrey and Powell (2002b), Godfrey and Powell (2002a), Powell and Topaloglu (2005), Topaloglu and Powell (2006), Powell et al. (2007b)). Godfrey and Powell (2002a) estimate separable nonlinear approximations of value functions with extensions to the multiperiod case in Godfrey and Powell (2002b). In Topaloglu and Powell (2006), tasks arriving over time have to be covered by resources of different type, and use a similar adaptive DP-based formulation to Godfrey and Powell (2002b), which makes use of linear or nonlinear approximations of the value function. Powell and Topaloglu (2005) provide some more transportation examples that this approach can be applied to. However, the problems considered in these papers are different to the problems faced by tramp shipping brokers, and use approximation methods rather than providing provably optimal algorithms as we provide in this setting. Another approach from Carlsson and Delage (2013) is to assign vehicles (ships) to a region, and only serve customers (cargoes) from that region. While this approach has advantages in terms of ease of implementation, it is restrictive in the shipping setting. There also exist robust approaches to model the uncertainty in vehicle routing (Agra et al. (2013), Kong et al. (2010)), but these tend to be too conservative for the spot market.

Another related stream of literature is the work by Powell (1987), Powell et al. (1988) with applications in long haul trucking. However, there are several key differences with our chapter. Powell et al. (1988) consider a heuristic approach to solve a scenario with multiple vehicles, while we consider an optimal algorithm to solve

the case of only one vehicle. In the shipping industry that this work is motivated from, we argue that there is little benefit in modeling the extra complexity of the multi-vehicle scenario because in practice there is little overlap between cargoes offered to ships under the same management. This occurs because the tramp shipping industry is characterized by many players, each with a relatively small market share, and the geographic dispersement of ships. As a result, each ship can be considered independently. This allows us to solve our model to optimality rather than using an approximation method. The formulation from Powell et al. (1988) is solved as a network flow problem with two kinds of arcs: deterministic arcs for known cargoes in the current period and stochastic arcs which model the value of having a vehicle at a location by considering the cargoes that could be available probabilistically. Additional future effects are captured using averages of historical earnings from given locations. This is in effect a two stage stochastic optimization model, whereas our model is a multistage stochastic optimization problem. Another key difference between trucking and tramp shipping problems is the degree of uncertainty in future prices. Due to the fact that the distances travelled by ships are often much greater than trucks, and because there is inherent volatility in the shipping market, the prices for cargoes can change significantly between subsequent voyages. This is in contrast to the trucking market where the prices are relatively stable over the next few trips, so the emphasis is on modeling the uncertainty in cargo availability rather than prices. We focus on this difference in Section 5.7. Another differentiating factor from the literature is the treatment of the sets of available cargoes. While many online optimization problems only consider items that arrive one at a time, we consider the scenario where we have an uncertain set of items to choose from at each time period, giving the problem a combinatorial structure. For a comprehensive overview of dynamic pickup and delivery problems, we direct the reader to the review paper of Berbeglia et al. (2010) and the references therein.

5.1.3 Contributions

Our main contributions are as follows:

- We propose a Markov Decision Process (MDP) model for the tramp shipping problem that captures the dynamic and stochastic nature of spot cargo availability. We also capture how the brokers consider potentially large sets of cargoes in each period. To the best of our knowledge, we are the first to model a tramp shipping firm that operates only in the spot market.

- Unfortunately, in general, our proposed formulation is hard to solve using a traditional dynamic programming approach. This is due to the state space of the problem being exponential in the number of cargoes considered. We propose a novel methodology for solving this MDP in a tractable way, by introducing a ranking algorithm which is polynomial in the number of possible cargoes at each port. We prove this algorithm is equivalent to solving the proposed DP.

- Through real-world data, we build a simulation environment which allows us to evaluate the performance of this algorithm relative to several other MDP approximation methods. We show that our ranking algorithm outperforms several benchmarks and the average performance of ships operating on the spot market in practice by between 4% and 32% depending on the number of cargoes the broker is able to consider.

- For the case of independent shipping rates, and when these shipping rates (also referred to as prices) are discrete, we propose an extension of the ranking algorithm. We show that this algorithm is polynomial in terms of the number of cargoes and discrete rate levels.

- We show that we can extend the ranking algorithm to be applied in the infinite horizon problem using linear programming or alternatively a modified policy iteration algorithm. Not only is this interesting from an analytical perspective but it is also a practical objective for ship brokers to optimize, since the solution to the infinite horizon problem presents a steady state view of the system which is useful to the shipping company for medium term planning.

- The models we develop also apply to a wider range of logistics problems. For

instance, oversubscribed trucking, taxi or ride sharing companies (such as Uber) face the problem of deciding which customers (these correspond to cargoes in this chapter) to accept or reject. The algorithms we propose can be applied to this setting given data on the availability of customers with different values at subsequent time periods. This algorithm can also be applied to other areas of service operations. As an example, suppose we are a firm who has to decide which jobs to take over a finite time horizon, where we are not certain about which jobs we will observe in the future, but can estimate (from data or otherwise) the probability that those jobs will appear. These jobs can have heterogenous times, rewards and durations.

5.2 Deterministic shipping rate model

We begin with deterministic shipping rates and then extend to uncertain shipping rates in order to isolate one difficulty at a time. We note that the deterministic shipping rate model is operationally useful due to the existence of financial instruments known as Forward Freight Agreements (FFAs). These are tools shipping companies use in practice to hedge against future uncertainty in shipping rates and can be thought of as the market expectation of what shipping rates will be in the future. If the brokers invest in FFAs in advance, they are essentially setting the price in the future to be what the FFA rate is today, making the prices deterministic and allowing the application of the model and algorithm we study below.

We model this problem using a Markov Decision Process (see for example Bertsekas (1995)). A simple example illustrating this model is given in Appendix C.2. This will help the reader parse the notation. Define a tuple of states $(i, t, \mathcal{S})$, where $i \in \{1 \dots N\}$ is the port where the ship is located at time $t \in \{1, ..., T\}$, and $\mathcal{S}$ is the set of observed cargoes the ship has to choose from. There is a finite time horizon consisting of T intervals. The set of all possible cargoes is denoted by $\mathcal{C}$. Because each cargo can either be available or not at time t, there are $2^{|\mathcal{C}|}$ different possible observations of cargo $\mathcal{S}$. Our decision, based on the state of the system is

$\mu(i,t,\mathcal{S}) = u \in \mathcal{S}$. This represents the cargo the broker chooses to pick up. Over a time horizon, the sequence of state dependent decisions is known as the policy $\pi = \{\mu(i,1,\mathcal{S}), \mu(i,2,\mathcal{S}), ..., \mu(i,T,\mathcal{S})\}$. Denote $G(i,u,t)$ as the reward associated with choosing cargo u from port i at time t. This is a deterministic function of the distance for the voyage, and the shipping rates at the time the cargo is chosen. The rates are assumed to follow a known forecast. In section 5.7 we consider the case where rates are uncertain. We define $D(i,u)$ to be the time taken to arrive at discharge port $d(u)$ if we choose port u from port i.

We assume each cargo i has an independent probability p_i of being available in each time period. This independence applies between cargos and time periods. This is consistent with practice (as an example, see data in Appendix C.3). In addition, we present some intuition motivated by our collaboration with the shipping company in order to explain why this is a valid assumption for this setting. First, due to the relatively large distances covered by ships in this industry, decisions about which cargo to pick are made on average 10 days apart. While day-to-day cargo independence may be viewed by some as a strong assumption, when decisions are separated by a significant time interval (which is the case in this setting) this is realistic. Second, the ships are likely to discharge in a different region from where they picked up the cargo. This means that the cargoes under consideration between two time periods are unlikely to overlap. The independence assumption between cargoes within the same time period is not imposed for model tractability, but rather makes the solution considerably simpler to describe and makes imputation from data more straightforward. It is possible to have any joint distribution on cargo availability, but for simplicity we will assume they are independent for the remainder of the chapter. Therefore, the probability of observing cargo availability set $\mathcal{S}$ is $P(\mathcal{S}) = \prod_{j\in\mathcal{S}} p_j \prod_{j\notin\mathcal{S}}(1-p_j)$ due to the independence property between cargoes.

The transition function is as follows: if we choose cargo u, the system transitions from state $(i,t,\mathcal{S})$ to $(d(u), t + D(i,u), \mathcal{S}')$. We make our next decision at the destination of the cargo $d(u)$ at time period $t + D(i,u)$ after observing a new set of cargoes. However, for notational simplicity, we will simply denote this future state

by $f(u,i)$. Naturally, we can use dynamic programming to solve this problem. If we define $J(i,t,\mathcal{S})$ to be the expected remaining profit over the remaining time horizon from time t, given the location of the ship and available cargoes, we can form Bellman's equations at every state:

$$\begin{aligned} J(i,t,\mathcal{S}) = &\max_{u\in\mathcal{S}}\{G(i,u,t) + \mathbf{E}_{\mathcal{S}}J(f(u,i),\mathcal{S})\} \\ &\forall t \in \{1\dots T\},\ \forall i \in \{1\dots N\},\ \forall \mathcal{S} \subset \mathcal{C} \end{aligned} \tag{5.1}$$

The initial conditions are $J(i,T,\mathcal{S}) = 0, \quad \forall i \in \{1\dots N\},\ \forall \mathcal{S} \subset \mathcal{C}$ denoting that there is no residual profit once we reach the end of the time horizon. It is well known (Bertsekas (1995)) that if we solve these equations using backwards recursion, we find the optimal policy. Note how these equations capture the intuition that the long term benefit of a cargo can be decomposed into the immediate profit $G(i,u,t)$ of taking the voyage, and the residual profit $\mathbf{E}_{\mathcal{S}}[J(f(u,i),\mathcal{S})]$ associated with ending up at a discharge port for the start of the next voyage. This can be equivalently represented as

$$V(i,t) = \mathbf{E}_{\mathcal{S}}[\max_{u\in\mathcal{S}}\{G(i,u,t) + V(f(i,u))\}]$$

by using the substitution $V(i,t) = \mathbf{E}_{\mathcal{S}}J(i,\mathcal{S},t)$. This can be thought of as the value associated with being at port i without the knowledge of which cargoes are available and is known in the literature as a "post-decision state".

This approach reduces the state space from $NT2^{|\mathcal{C}|}$ to NT, so the look-up table we have to access to can be much smaller. However, we still have to calculate the expectation with respect to $\mathcal{S}$ at every port, which requires $2^{|\mathcal{C}|}$ calculations and is computationally intractable.

5.3 Ranking algorithm

In what follows, we develop a ranking based algorithm that solves the Markov Decision Process efficiently. In particular, we show that the ranking algorithm below is polynomial in the number of possible cargoes rather than exponential. This leads

to a tractable algorithm for solving the problem under consideration. The intuition behind this reduction is that the brokers don't need to know the availability of all cargoes to make a decision, just what the most valuable (highest ranked) available cargo is. If we rank all the cargoes and take the expectation with respect to this random variable, this is more computationally efficient than taking the expectation with respect to all possible cargo combinations. We will prove that this algorithm is equivalent to the DP outlined in the previous section, and is therefore optimal for solving the Markov Decision Problem. We refer the reader to Appendix C.3 for a step by step example of how this ranking algorithm works.

We now describe the ranking algorithm in detail:

Algorithm 6 Ranking algorithm

Initialize with $V(i,T) = 0, \quad \forall i = 1, ..., N.$

$r = 0$, $i =$ randomly assigned start port

for each time step $t = T-1, ..., 1$ **do**

    for each state $i = 1, ..., N$ **do**

        $Q(i,u,t) = G(i,u,t) + V(f(i,u)), \ \forall u \in \{1, 2, ..., C\}$

        Sort $Q(i,u,t)$ from largest to smallest to get a ranking $s_1, s_2, ..., s_N$

        Calculate $V(i,t) = \sum_{h=1}^{N} \left[G(i,u,t) + V(f(i,u))\right] p_{s_h} \prod_{j=1}^{h-1}(1 - p_{s_j})$

The ranking algorithm requires $O(TN|\mathcal{C}|^2)$ calculations. At every port and time (TN combinations), $|\mathcal{C}|\log(|\mathcal{C}|)$ calculations are required to rank the possible cargoes, while $|\mathcal{C}|^2$ calculations are required to calculate the expectation over which cargoes will be available. This is much better than the $O(2^C)$ required for the previous DP approach.

5.3.1 Ranking algorithm over polymatroids

In what follows, we present Theorem 7 showing that the ranking algorithm is optimal. We do this by showing that solving Bellman's equation with regards to the post decision state

$$V(i,t)^{DP} = \mathbf{E}_{\mathcal{S}}[\max_{u \in \mathcal{S}}\{G(i,u,t) + V(f(i,u))\}] \qquad \text{(DP(i,t))}$$

is equivalent to solving the following Linear Program:

$$
\begin{aligned}
V(i,t)^{LP} = \text{maximize}_y \quad & \sum_{u \in \mathcal{C}} y_u (G(i,u,t) + V(f(i,u))) & & \\
\text{s.t} \quad & \sum_{u \in \mathcal{S}} y_u \leq P_A(\mathcal{S}) & \forall \mathcal{S} \subseteq \mathcal{C} \qquad & \text{(PLP(i,t))} \\
& \sum_{u \in \mathcal{C}} y_u = P_A(\mathcal{C}) & &
\end{aligned}
$$

where y_u can be interpreted as the probability we choose cargo u (note this is different from the probability that cargo u is available, because it is influenced by our decision), and $P_A(S)$ is the probability that at least one cargo in the set S is available. Note, $V(f(i,u))$ is assumed to be constant in the equations for $V^{LP}(i,t)$ and $V^{DP}(i,t)$, as this is the value for a future time period and is assumed to have already been calculated via backwards recursion.

Theorem 7. *(i) $V^{LP}(i,t) = V^{DP}(i,t)$, (ii) under the assumption of cargo independence across time periods, the ranking policy is optimal*

We show the proof in Section 5.3.2. In what follows we first provide some intuition. PLP(i,j) has special structure, because the feasible region is a polymatroid. Edmonds (1970) proved that such an LP can be solved using the greedy algorithm, and gives a closed form expression for the variables. Suppose we are trying to solve $\max\{\sum_{i=1}^{n} c_i y_i : y \in \text{polymatroid}\}$, and we order the terms in the objective by an ordering $s_1, ..., s_n$, such that $c(s_1) \geq c(s_2) \geq ... \geq c(s_n)$ (where $c(s_i)$ refers to the objective coefficient corresponding to variable y_{s_i}). Then following Edmonds' result, the solution is given by (see Lemma 6 for more details):

$$
\begin{aligned}
y_{s_1} &= P_A(\{s_1\}) = p_{s_1} \\
y_{s_i} &= P_A(\{s_1, ..., s_i\}) - P_A(\{s_1, ..., s_{i-1}\}) = p_{s_i} \prod_{j=1}^{i-1} (1 - p_{s_j})
\end{aligned}
$$

This confirms the intuition that the probability we choose the i^{th} ranked cargo, is the probability that the i^{th} ranked cargo is available and no higher ranked cargo is

available. We can recover the expected value of being in a state by multiplying the probabilities of choosing a cargo, with the value of that cargo.

$$V(i,t) = \sum_{h=1}^{|\mathcal{C}|} \left[G(i,s_h,t) + V(f(i,s_h))\right] p_{s_h} \prod_{j=1}^{h-1}(1-p_{s_j})$$

We can then solve for the entire set of ports over the time horizon, by starting at time zero with the necessary boundary condition, and working backwards through time, calculating $V(i,t)$ for every port.

5.3.2 The equivalence between Bellman's equation and the polymatroid LP

In this section we prove the main result of this chapter. This proof formalizes the intuition discussed in Section 5.3.1.

We begin by identifying some features of the policy that we will use in the proof. We can define the probability of taking different cargoes given a policy μ. In a policy, we will make decision u in response to cargo availability S, if $\mu(S) = u$. The probability we choose cargo u is given by $y_u = \mathbf{E}_S[1_{\mu(S)=u}] = \sum_{S|\mu(S)=u} P(S)$. The value of following a policy is clearly $V_\mu(i,t) = \sum_u^{|\mathcal{C}|} y_u(\mu)\left[G(i,u,t) + V(f(i,u))\right]$, which is the probability of each action times the reward.

Definition 5. *A* **ranking policy** *is a policy where we serve cargoes in the order of a priority list $s_1, ..., s_n$. This means that we only take cargo s_i when cargoes $s_1, .., s_{i-1}$ are not available. In terms of an observation of cargoes S, the decision we make is $\mu(S) = \min\{i|s_i \in S\}$.*

We can calculate the probability we will take an action according to a ranked policy by taking the expectation with regards to the observed cargoes. The probability that we take action s_i is the probability associated with the event that cargoes $s_1, .., s_{i-1}$ are not available, but s_i is.

$$y_{s_i} = P(\{s_i \in S\} \cap \{s_1, .., s_{i-1} \notin S\}) = P(A_{s_i}) \prod_{u \in \{s_1,...,s_{i-1}\}} (1 - P(A_u)) = p_{s_i} \prod_{u \in \{s_1,...,s_{i-1}\}} (1 - p_u) \tag{5.2}$$

Now let us show how to construct the polymatroid (PLP(i,t)). Let A_u be the event that cargo u is available, so that $P(A_u) = p_u$. Define $P_A(S)$ to be the probability that at least one cargo in the set S is available. This probability is the compliment to the probability that no cargoes in the set are available, $P_A(S) = P(\cup_{u \in S} A_u) = 1 - P(\cap_{u \in S} A_u) = 1 - \prod_{u \in S} p_u$.

$P_A(\cdot)$ is nondecreasing (if we add a cargo to the set, the probability at least one of the cargoes is available will increase) and furthermore, we can show it is submodular.

Lemma 4. *$P_A(\cdot)$ is a submodular set function*

Proof. Let $S^{(1)} \subseteq S^{(2)} \subseteq \mathcal{C}$. Let $\{u\} \in \mathcal{C} \setminus S^{(1)}$.

$$\begin{aligned} P_A(S^{(2)} \cup \{u\}) - P_A(S^{(2)}) &= P_A(S^{(2)}) + P_A(\{u\}) - P_A(S^{(2)} \cap \{u\}) - P_A(S^{(2)}) \\ &= P_A(\{u\}) - P_A(S^{(2)} \cap \{u\}) \\ &\leq P_A(\{u\}) \\ &= P_A(\{u\}) - P_A(S^{(1)} \cap \{u\}) \\ &= P_A(S^{(1)} \cup \{u\}) - P_A(S^{(1)}) \end{aligned}$$

□

where in the fourth equality $P_A(S^{(1)} \cap \{u\}) = 0$ since $\{u\} \in \mathcal{C} \setminus S^{(1)}$).

Definition 6. *Let $f : 2^{\mathcal{C}} \to R_+$ be a nondecreasing, submodular set function. The polyhedron $\{\mathbf{x} \in R^{|\mathcal{C}|} | \sum_{i \in S} x_i \leq f(S),\ \forall S \subset C; \sum_{i \in \mathcal{C}} x_i \leq f(\mathcal{C})\}$ is a polymatroid.*

If we define $\mathcal{P} = \{\mathbf{y} \in R^{|\mathcal{C}|} | \sum_{u \in S} y_u \leq P(S),\ \forall S \subseteq \mathcal{C}, \sum_{u \in \mathcal{C}}, y_u = P_A(\mathcal{C})\}$, then clearly this is a polymatroid. We next show that for any valid policy, the probability that each cargo is accepted lies within the polymatroid $\mathcal{P}$.

Lemma 5. *For any admissible policy μ where $y_u = \mathbf{E}_{\mathcal{S}}[1_{\mu(\mathcal{S})=u}]$, $y \in \mathcal{P}$.*

Proof. Let B_u be the event that cargo u is chosen under policy μ. As these events are disjoint, we have $P(\cup_{u\in S} B_u) = \sum_{u\in S} y_u,\ \forall S \subset C$. Clearly $B_u \subseteq A_u$ because a cargo can only be taken if it is available, so that $P_A(S) = P(\cup_{u\in S} A_u) \geq P(\cup_{u\in S} B_u) = \sum_{u\in S} y_u,\ \forall S \subset C$. Finally, it follows that a cargo is taken whenever a cargo is available, so that $P_A(S) = \sum_{u\in C} y_u$. This establishes that $\mathbf{y}$ is feasible for the polymatroid. □

We will next show that for any extreme point of the polymatroid, there exists a corresponding valid policy.

Lemma 6. *For any $y \in ext(\mathcal{P})$, there exists a policy μ such that $y_u = \mathbf{E}_{\mathcal{S}}[1_{\mu(\mathcal{S})=u}]$.*

Proof. Let $\hat{y}$ be an extreme point of the polymatroid. From polymatroid theory (Edmonds (1970)), we know that this can be characterized by an ordering of variables $s_1, ..., s_n$, such that

$$
\begin{aligned}
\hat{y_{s_1}} &= P_A(\{s_1\}) = p_{s_1} \\
\hat{y_{s_i}} &= P_A(\{s_1, ..., s_i\}) - P_A(\{s_1, ..., s_{i-1}\}) \\
&= P(\cup_{u\in\{s_1,...,s_i\}} A_u) - P(\cup_{u\in\{s_1,...,s_{i-1}\}} A_u) \\
&= P(\cup_{u\in\{s_1,...,s_{i-1}\}} A_u) + P(A_{s_i}) - P(\cup_{u\in\{s_1,...,s_{i-1}\}} A_u \cap A_{s_i}) - P(\cup_{u\in\{s_1,...,s_{i-1}\}} A_u) \\
&= P(A_{s_i}) - P(\cup_{u\in\{s_1,...,s_{i-1}\}} A_u \cap A_{s_i}) \\
&= P(A_{s_i})(1 - P(\cup_{u\in\{s_1,...,s_{i-1}\}} A_u)) \\
&= P(A_{s_i}) P(\neg \cup_{u\in\{s_1,...,s_{i-1}\}} A_u) \\
&= P(A_{s_i}) P(\cap_{u\in\{s_1,...,s_{i-1}\}} \neg A_u) \\
&= P(A_{s_i}) \prod_{u\in\{s_i,...,s_{i-1}\}} (1 - P(A_u)) \\
&= p_{s_i} \prod_{j=1}^{i-1} (1 - p_{s_j})
\end{aligned}
$$

This is equivalent to a ranking policy from (5.2). Therefore we have that $\hat{y_u} = \mathbf{E}_{\mathcal{S}}[1_{\mu(\mathcal{S})=u}]$ for some μ. □

Proof of Theorem 7: (i) For any admissible policy μ, the reward is $V_\mu(i,t) = \sum_u^{|\mathcal{C}|} y_u(\mu)\,[G(i,u,t) + V(f(i,u))]$, with $V^{DP}(i,t) = \max_\mu V_\mu(i,t)$. This is the same as the objective for PLP(i,t). From Lemma 5, $y_u(\mu)$ from any admissible policy is feasible for the polymatroid $\mathcal{P}$, so PLP(i,t) is a relaxation of DP(i,t). Therefore $V^{DP} \leq V^{LP}$.

Furthermore, because the feasible region $\mathcal{P}$ is a polyhedron, there exists an optimal solution to PLP(i,t) at an extreme point y^{LP}. From Lemma 6 there is a corresponding admissible policy which results in $y^{LP} = \mathbf{E}_{\mathcal{S}}[1_{\mu^{LP}(\mathcal{S})=u}]$, so that $V_{\mu^{LP}} = V^{LP}$. It follows that $V^{DP} = V^{LP}$.

(ii) The proof of this result follows by induction; if we make the same decision at each stage as the DP, then the ranking policy is optimal. $\square$

5.4 Infinite horizon

In what follows, instead of a finite horizon problem, we will consider the analogous discounted infinite horizon problem. This a valid practical objective for ship brokers to want to optimize, since the solution to the infinite horizon problem presents a steady state view of the system. This is more useful to the shipping company for medium term planning than the finite horizon formulation where we might get distortions in behavior towards the end of the time horizon. More broadly, it is also interesting from an analytical perspective on how to make decisions from an uncertain discrete set of options over time. We show that we can extend the ranking algorithm to be applied in the infinite horizon problem also, resulting in significant gains in efficiency over standard policy iteration or LP reformulation techniques. In the infinite horizon case, the problem we would like to solve becomes

$$\max_\pi \mathbf{E}_\pi[\sum_{t=1}^{\infty} \beta^{D(i,\mu(i,\mathcal{S}))} G(i,\mu(i,\mathcal{S}))]$$

Bellman's equation can be formed using the post decision state:

$$V(i) = \mathbf{E}_{\mathcal{S}}[\max_{u \in \mathcal{S}}\{G(i,u) + \beta^{D(i,u)}V(f(i,u))\}], \ \forall i \in \{1,..,N\} \tag{5.3}$$

Note how we have used $\beta^{D(i,u)}$ as the discount factor, where $D(i,u)$ is the duration of the voyage if we pick up cargo u from port i. We will present two different algorithms for solving this problem, one a variant of the well known policy iteration algorithm, which we will call rank and iterate, and one which solves an LP that evolves from our polymatroid formulation with global constraints linking the value functions for separate ports.

The crux of each of these methods is that solving Bellman's equation at each state is equivalent to solving an LP with polymatroid constraints. Indeed, applying Theorem 7, if we define T to be the Bellman operator such that $T\mathbf{V} = \mathbf{E}_{\mathcal{S}}[\max_{u \in \mathcal{S}}\{G(i,u) + \beta^{D(i,u)}V(f(i,u))\}] \ \forall i \in \{1,..,N\}$, where $\mathbf{V} = \{V(i)\}_{i=1}^{N}$, then solving the following LP is equivalent to applying the Bellman operator:

$$V(i) = \text{maximize}_{\mathbf{y}} \quad \sum_{u \in \mathcal{C}} y_{iu}(G(i,u) + \beta^{D(i,u)}V(f(i,u))) \tag{5.4}$$

$$\text{s.t} \quad \sum_{u \in \mathcal{S}} y_{iu} \leq P_A(\mathcal{S}) \qquad \forall \mathcal{S} \subseteq \mathcal{C} \tag{5.5}$$

$$y_{iu} \geq 0 \tag{5.6}$$

As a result, we can leverage the rich theory of dynamic programming (Puterman (2014), Bertsekas (1995)) for methods to extend the ranking algorithm to the infinite horizon.

5.4.1 Rank and iterate

Recall that in policy iteration, given an initial set of value functions $\mathbf{J}$, we can iterate between the two steps until we reach a convergence criterion:

1. find a policy $\boldsymbol{\mu}$ such that

$$\mu(i, \mathcal{S}) = \arg\max_u \{G(i,u) + \beta^{D(i,u)} \mathbf{E}_{\mathcal{S}} J(f(i,u))\},\ \forall i \in \{1...N\},\ \forall \mathcal{S} \subseteq \mathcal{C}$$

2. solve the system of equations

$$J(i, \mathcal{S}) = G(i, \mu(i, \mathcal{S})) + \beta^{D(i,\mu(i,\mathcal{S}))} \mathbf{E}_{\mathcal{S}} J(f(i, \mu(i, \mathcal{S}))),\ \forall i \in \{1...N\},\ \forall \mathcal{S} \subseteq \mathcal{C}$$

From Bertsekas (1995), we know that the policy iteration algorithm will converge to the optimal policy in a finite number of iterations. Motivated by this we propose the following algorithm.

Algorithm 7 Ranking algorithm infinite horizon

Start with an initial set of value functions $\mathbf{V} = \{V(i)\}_{i=1}^N$.
while Value functions haven't converged **do**
    for $i \in \{1...N\}$ **do**
        for $u \in \{1...N\}$ **do**
            Calculate probability of taking each action u according to ranking y_{iu}:
            $Q(i,u,t) = G(i,u,t) + V(f(i,u)),\ \forall u \in \{1,2,...,C\}$
            Sort $Q(i,u,t)$ to get a ranking $s_1, s_2, ..., s_N$
            Calculate $y_{is_h} = p_{s_h} \prod_{j=1}^{h-1}(1 - p_{s_j}),\ \forall h \in \{1,2,...,C\}$
    Solve the system of equations:

$$V(i) = \sum_{u \in \mathcal{C}} y_{iu}(G(i,u) + \beta^{D(i,u)} V(f(i,u))),\ \forall i \in \{1...N\}$$

This works because, for a given set of value functions, finding the policy μ which maximizes the value of each state, is equivalent to finding the probabilities y_{iu} a cargo is taken. This in turn maximizes each state from Theorem 7. In the second step, solving the system of equations that enact this policy is also equivalent. Therefore, because the updates being made in each iteration are the same, the rank then iterate

algorithm will also converge in a finite number of iterations.

This algorithm has an advantage over standard policy iteration. Rather than having to calculate a policy with respect to all $2^{|\mathcal{C}|}$ possible cargo combinations, we can instead utilize our ranking procedure which is polynomial in the number of cargoes considered. This causes substantial savings in computation time in each iteration.

5.4.2 Infinite horizon ranking LP

It is well known that it is possible to solve an infinite horizon DP (Puterman (2014), De Farias and Van Roy (2003)) using the following formulation,

$$\text{minimize}_{\mathbf{V}} \quad c\mathbf{V} \tag{5.7}$$

$$\text{s.t} \quad T\mathbf{V} \leq \mathbf{V} \tag{5.8}$$

Where c is any strictly positive vector, and $\mathbf{V} = \{V(i)\}_{i=1}^{N}$ is the vector of value functions. We can reformulate this using our polymatroid representation of the Bellman operator as follows:

$$\text{minimize}_{\mathbf{V}} \sum_{i} V(i) \tag{5.9}$$

$$V(i) \geq \quad \text{maximize}_{y_i} \sum_{u \in \mathcal{C}} y_{iu}(G(i,u) + \beta^{D(i,u)} V(f(i,u))) \qquad \forall i \in \{1,..,N\} \tag{5.10}$$

$$\text{s.t} \quad \sum_{u \in \mathcal{S}} y_{iu} \leq P_A(\mathcal{S}) \qquad \forall \mathcal{S} \subseteq \mathcal{C}, \quad \forall i \in \{1,..,N\} \tag{5.11}$$

$$y \geq 0 \tag{5.12}$$

However, we notice that this formulation is difficult to solve because of the non-linearities in constraint (5.10). We can overcome this by dualizing each inner problem

for port i as follows:

$$\text{minimize} \sum_i V(i) \tag{5.13}$$

$$\text{s.t} \qquad V(i) \geq \sum_{\mathcal{S} \subseteq \mathcal{C}} w_{i\mathcal{S}} P_A(\mathcal{S}) \qquad \forall i \in \{1, .., N\} \tag{5.14}$$

$$\sum_{\mathcal{S}|u \in \mathcal{S}} w_{i\mathcal{S}} \geq (G(i,u) + \beta^{D(i,u)} V(f(i,u))) \qquad \forall u \in \mathcal{C}, \quad \forall i \in \{1, .., N\} \tag{5.15}$$

This has an advantage over the standard LP formulation for solving the DP because our formulation has $O(2^{|C|}N)$ variables and $O(|C|N)$ constraints, as opposed to the standard LP formulation which has $O(2^{|C|}N)$ variables and $O(2^{|C|}|C|N)$ constraints. However, the large number of variables means that in practice, the rank then iterate algorithm performs better for large problem instances. The infinite horizon LP has the advantage that we know exactly the problem size, and if we have a good idea of the speed of the LP solver we should be able to estimate the run time requirements. Although in practice this isn't usually the case, the number of iterations the rank then iterate algorithm needs to converge could be very large, in the worst case, equal to the number of different policies.

5.5 Perfect information benchmark

In the literature, in order to determine how good an approximation algorithm is in the presence of uncertainty, it is common to consider as a benchmark the perfect information setting. Suppose we know in advance which cargoes will be available over the time horizon. We can then solve a simple deterministic optimization problem to find the maximum profit that could have been obtained if we had this information ahead of time. This provides an upper bound on the performance of the stochastic dynamic programming problem which incorporates the limited knowledge ship brokers have available. As a result, this provides a good benchmark to compare with.

The shipping network can be modeled as a directed graph where the vertices $\mathcal{V}$ are ports. To capture that our decisions are made over a finite horizon of T days, we

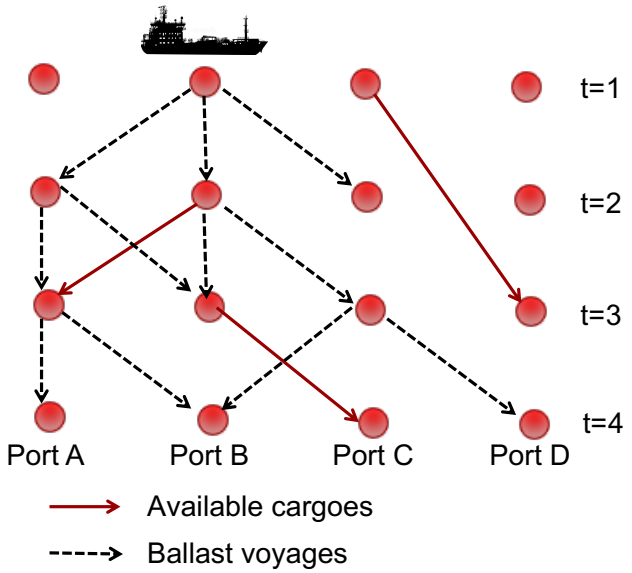


Figure 5-2: An example network for the perfect information problem

have a vertex for each port at every time t. If $\mathcal{V}_t$ is the set of ports at time t, then $\mathcal{V} = \cup_{t=0}^{T} \mathcal{V}_t$. The arcs $\mathcal{A}$ are possible shipping routes between ports. Suppose we know all cargoes that will be available over the time horizon. If a cargo departs from port i to j at time t, there is an arc from port/node i at time t to port/node j at time $t + D_{ij}$, where D_{ij} is the time taken to travel from port i to port j. Let G_{ijt} be the reward associated with this cargo. Ships can also travel between ports without carrying a cargo (such as traveling on a ballast leg in order to arrive at a port to load a cargo), making the graph connected. In this case the arc would have G_{ijt} negative to reflect the cost involved in ballast voyages. With the network constructed above, let $I(jt)$ be the set of incoming arcs to port j at time t, while $O(jt)$ is the set of outgoing arcs from port j at time t, known a priori.

Our decision variable x_{ijt}, is 1 if the ship travels from port i to j at time t, and 0 otherwise. We can formulate the following integer optimization problem to find the optimal sequence of cargoes for a ship.

$$\text{maximize} \quad \sum_{(ijt)\in\mathcal{A}} G_{ijt} x_{ijt} \tag{5.16}$$

$$\text{s.t} \quad \sum_{i\in I(jt)} x_{ijt} = \sum_{k\in O(jt)} x_{jkt} \qquad \forall i \in \mathcal{V} \tag{5.17}$$

$$\sum_{j \in O(i1)} x_{ij1} = 1 \qquad \text{if there is a ship starting at port i} \tag{5.18}$$

$$\sum_{j \in \mathcal{V}} \sum_{i \in I(jT)} x_{ijT} = 1 \tag{5.19}$$

$$x_{ijt} \in \{0, 1\} \qquad \forall (i, j) \in \mathcal{S}, \quad t \in \{1, 2, ..., T\} \tag{5.20}$$

The first constraint is a flow conservation constraint. The second constraint initializes a unit flow at the current port and time and the final constraint collects the unit flow from any possible port at the end of the time horizon. This formulation is naturally integer (can be reformulated as an uncapacitated network flow problem with integer inflows/outflows), so the integrality constraints can be relaxed as the underlying constraint matrix is totally unimodular. As a result, this model can be solved efficiently as a linear program.

This is equivalent to solving a number of longest path problems between the current port and all possible ports at the end of the time horizon, where the length of an arc corresponds to the profit obtained. Since the graph is a directed acyclic graph (all arcs represent journeys and therefore move forward in time) solving the longest path problem is equivalent to solving the shortest path problem. We could solve this using dynamic programming or a specialized algorithm for solving the shortest path problem (see for example Ahuja et al. (1993)), but this LP is sufficiently fast for the scale of problems considered in the shipping industry.

We generate the arcs in the graph corresponding to the probabilities that different cargoes exist. If we solve this problem repeatedly for sample paths of available cargoes, we can approximate the expected value of having perfect information.

5.6 Approximate benchmark methods

In the following section, we compare the ranking algorithm with several other heuristic methods that are often used in the literature. These can be viewed as alternatives or benchmarks. However, these methods have no guarantee of optimality.

One-Step-Lookahead (1SL) policy

In the one-step-lookahead model, the shipping broker chooses the cargo with the greatest average profit per day over the next voyage, ignoring all residual profit after this voyage is finished. This myopic policy can be thought of as a one-step lookahead policy with a cost to go of 0. The policy is:

$$\mu(i, \mathcal{S}, t)^{1SL} = \arg\max_{u \in \mathcal{S}} \frac{G(i, u, t)}{D(i, u)} \tag{5.21}$$

This policy has the advantage of being very fast to compute, because it does not depend on the actions taken at previous time periods. Furthermore, it can be implemented online, so we do not need to compute the policy ahead of time.

Threshold heuristic

We propose a simplified deterministic dynamic program in which we split the cargoes into two groups, cargoes we are likely to observe and those we are unlikely to observe. We can modify the dynamic program formulation of Section5.2 by assuming all the likely cargoes to occur, and none of the unlikely cargoes to occur, thus making the problem deterministic. More formally, we classify cargoes according to a threshold ρ. Recall each cargo i has an independent probability p_i of existing in each time period. If $p_i \geq \rho$ then cargo i belongs to the set $\mathcal{L}$ of cargoes we are likely to observe. We can formulate a deterministic dynamic program where at each stage we pick a cargo from the set $\mathcal{L}$, taking into account the immediate profit from taking the cargo, and the residual profit from the port we discharge at:

$$J(i,t)^{Thr} = \max_{u \in \mathcal{L}} \{G(i,u,t) + J(f(i,u))^{Thr}\} \qquad \forall t \in \{1 \dots T\},\ \forall i \in \{1 \dots N\} \tag{5.22}$$

with initial conditions $J(i,T)^{Thr} = 0, \quad \forall i \in \{1 \dots N\}$. Because this formulation is deterministic, it has a much smaller state space, and can be solved very fast, as observed in Section 5.6.2. For the special case where $\rho = 0$, this is the same as assuming that all cargoes will occur in all periods. In this case, the control $\mu(i,t)^{Thr}$

gives us the highest ranked (most valuable) cargo, if all cargoes were available. This is the version we implemented as this generally performed well and is easy to interpret.

Unlike the algorithms we have shown so far, this policy is under-prescribed in the sense that the cargo the policy suggests we pick up may not be available. In this case, we followed the heuristic that the ship will wait until the cargo suggested by the policy does become available. While this may not be the optimal strategy, it is easy for brokers to implement and performs relatively well as a heuristic, as shown in Section 5.6.2.

Rollout algorithm

This threshold policy can be extended into a rollout algorithm. The threshold algorithm produces a heuristic policy π^{Thr} which can used to evaluate $J(i,t)^{Thr}$ the value of following the threshold policy from time t to T from port i. We can then make a decision over the actual observed cargoes in the current stage by solving

$$J(i,t,\mathcal{S})^{RO} = \max_{u \in \mathcal{S}} \{G(i,u,t) + J(f(i,u))^{Thr}\} \tag{5.23}$$

which is very similar to equation (5.1), but using the heuristic policy cost to go J^{Thr} rather than the optimal cost to go in the maximization problem. Intuitively, we pick the most valuable cargo in this stage (out of the cargoes which are available $\mathcal{S}$) by looking at the immediate profit plus an estimate of the residual value of discharging in port $d(u)$. This estimate comes from the precomputed threshold policy.

As discussed in Bertsekas (1995), we expect the rollout policy to perform better than the heuristic policy, but worse than the policy obtained by solving the full dynamic programming problem. As a result it follows that $J \geq J^{RO} \geq J^{Thr}$, which is observed in the simulations in Section 5.6.2. Furthermore this is a commonly used benchmark in the Dynamic Programming community. In the next section we will explore how these heuristics perform, relative to the optimal solution and the current routing policy.

5.6.1 Numerical results

Simulation algorithm

To test the different policies and algorithms we discussed above, we developed a simulation environment that closely approximates the actual situation faced by the brokers in practice. We used real world shipping data that we obtained from our partner shipping company to calibrate the simulation environment and optimization algorithms.

In what follows, we provide more details on the simulation. First we calculate the policy from either the ranking algorithm or one of the offline heuristics. Starting at the current port of the ship, we randomly generate a set of available cargoes $\mathcal{S}$ with probability $\prod_{i\in\mathcal{S}} p_i \prod_{j\notin\mathcal{S}}(1-p_j)$. We then look up the decision from the precomputed policy table. We then get the reward from making the decision, and progress forward in time, then repeat the process until we reach the end of the time horizon. This is described in Algorithm 8 below.

To reduce variance, we tested the policy from each method on the same sequence of cargoes. Each sequence of cargoes over the time horizon is a trial. We repeated for 1000 different trials to get an average expected reward. The time horizon we used was 100 days which corresponds to the planning window the shipping company typically operates within.

Algorithm 8 Simulation procedure for evaluating performance of policies

Data: R, C, L, T
$r = 0$, $i =$ randomly assigned start port
while $t < T$ **do**
    randomly generate cargo $\mathcal{S}$ according to $P(\mathcal{S}) = \prod_{i\in\mathcal{S}} p_i \prod_{j\notin\mathcal{S}}(1-p_j)$
    choose an action u from policy $\mu(i, \mathcal{S}, t)$
    if $u =$wait **then**
        $t = t - 1$,
    else
        $r = r + R(l(u), d(u), t) - C(i, u)$, $t = t - D(i, u)$, $i = d(u)$

5.6.2 Simulation results

In this section we tested the algorithms we implemented using the simulation environment described in Section 5.6.1. We implemented the algorithms in the `Python` programming language. The experiments were run on a macbook with an Intel Core i5@2.4GHz with 8GB RAM, and all tests were limited to a single thread.

We look at an aggregate problem, where we try and decide which regions a ship should travel between. These regions are formed by aggregating ports and cargoes together. For instance, if a ship is at a port in the Mediterranean, the probability that a cargo is available in the Black Sea is the aggregation of cargo availability at all ports in the Black Sea. This is consistent with medium-term planning objectives and it can provide insights that can be easily understood by the brokers. For convenience, in these simulations we assume that the cargoes in any region can be accessed by ships in any other region so that the number of ports/regions accessible is equal to the number of cargoes considered ($N = C$). In practice, the cost of the ballast (empty) leg required to pick up cargo would make some of the longer journeys unattractive and therefore, unlikely to be chosen. We can easily model this by placing restrictions on which cargoes are accessible from different locations.

Our experiments show how the performance of the policies changes as the number of cargoes/regions the broker has to choose from increases. We examine 4 different scenaria for a ship broker:

1. A 5 region network, focused on Europe (Black Sea, the Baltic, the Mediterranean, UK, and the North Sea),

2. expand to a 10 region network by adding the Middle East and parts of Asia (by adding Arabian Gulf, China / Taiwan, Pakistan / West Coast India, Red Sea, South East Asia),

3. expand to a 15 region network (by adding Caribs, Bahamas, US Gulf, Korea / Japan, South Pacific),

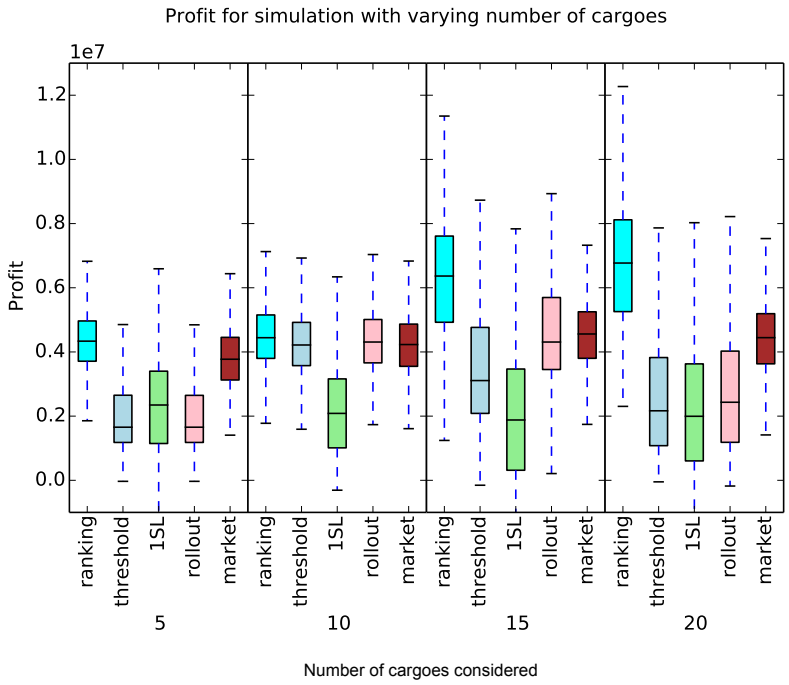


Figure 5-3: How the profit obtained by a broker changes as a function of the number of cargoes considered

4. expand to a 20 region network incorporating the major shipping regions around the globe.

We also compare against the current market performance. This is simulated using Markov Chain Monte Carlo (MCMC) methods, where the probability of picking up a cargo from a port given the ship is currently discharging at another port is estimated from historical data.

The simulation results are shown in Figure 5-3. Along the x-axis is each algorithm grouped by the number of cargoes considered in that scenario, while on the y-axis is the anonymized profit for each simulation. Notice how, on average, the ranking algorithm outperforms the other heuristic methods for any number of cargoes available. We also observe that the ranking algorithm outperforms the average performance of a ship on the spot market by at least 4% and up to 32%. The relative performance of the ranking algorithm, compared to other heuristics increases as the number of cargoes to choose from increases. Intuitively, this is because the ranking algorithm is able to use more complex policies which encompass not only the residual effects of

future voyages but also the full range of cargo availability scenarios, which becomes more rich as the problem size increases.

The market algorithm tends to perform better than the other simple heuristics we tried. This is probably due to the market's ability to have a good "sense" for what areas are favorable in the long run, compared to the other heuristics, as these tend to be more myopic. There is no clear next best amongst the remaining heuristics. The relative performance differs as we varied the number of cargoes we considered. As expected, the threshold algorithm is dominated by the rollout algorithm, but the margin of improvement is often small. The one-step-lookahead policy is myopic, and will often remain in a narrow geographic area even if there are gains to be made by incurring short term losses to travel to a more profitable area in the long term. This is a possible explanation for the lack of improvement with more cargoes. On the other hand, the threshold algorithm is highly sensitive to the availability of the most valuable cargo. If there is a valuable cargo that is rare, then the threshold cargo will wait until that cargo appears which is often a poor strategy. This is a possible explanation for the decrease in performance with the 20 region network. In summary, because of the issues we discussed for the different heuristics we considered, we believe the substantial increase in profitability justifies the small increase in computation required for the ranking algorithm. As shown in Table 5.1, even for a simulation with 20 cargoes to choose from, the ranking algorithm is still able to solve in seconds. Also, it is important to note that this simulation confirmed that the ranking algorithm is equivalent to the full dynamic program as proven in Theorem 7.

Comparison with perfect information

In Figure 5-4 , we observe how the ranking algorithm compares to the perfect information algorithm. The perfect information algorithm assumes brokers know which cargoes will appear over the time horizon a priori. Again we repeat the simulations over different problem instances using a different number of cargoes available. We observe that the ratio between the two methods stays relatively constant as the number of cargos is varied. The value of information by knowing what cargoes will appear in

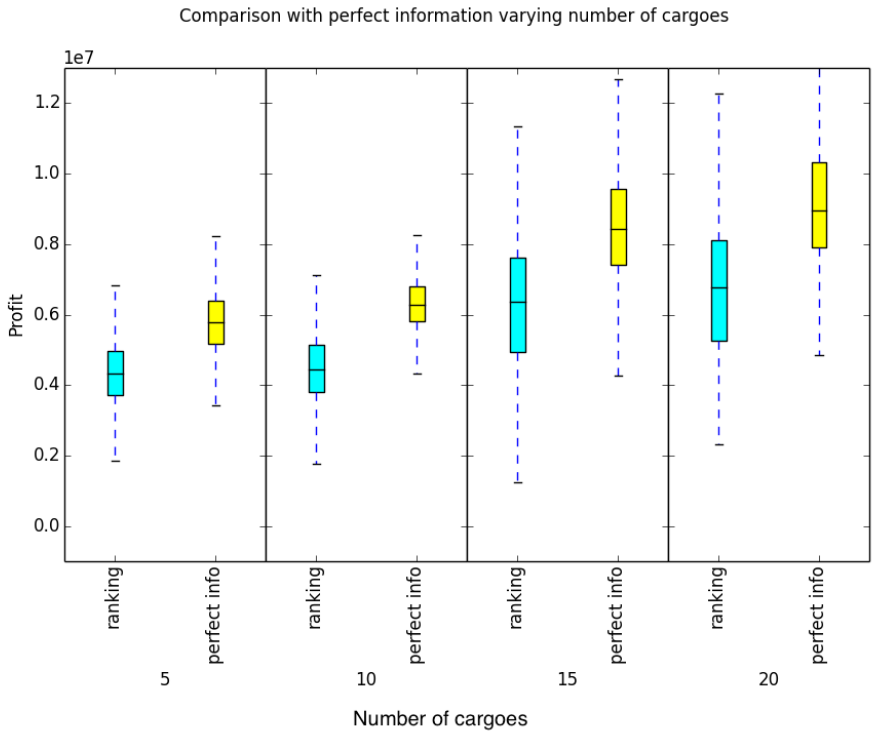


Figure 5-4: Number of cargoes considered perfect information comparison

advance is between 22-24% on average depending on the number of cargoes available.

Run-time comparison

Table 5.1: Comparison of run-time for differing numbers of cargoes

Policy	Complexity	Time taken (sec)				
		C = 5	8	10	20	40
Full DP	$O(TNC(2^C)^2)$	0.173	45.10	-	-	-
Ranking	$O(TNC^2)$	0.010	0.056	0.131	2.113	30.85
Threshold	$O(TNC)$	0.0035	0.021	0.040	0.331	2.398

In Figure 5-5 and Table 5-5 we compare runtimes between the different methods. To obtain these run times, we averaged 100 different trials for each problem size. In addition, we had a run time threshold of 5 minutes, to prevent excessively long run times. As expected, the full DP runtime exhibits an exponential trend as the number of possible cargoes increases, becoming quickly impractical as we approach

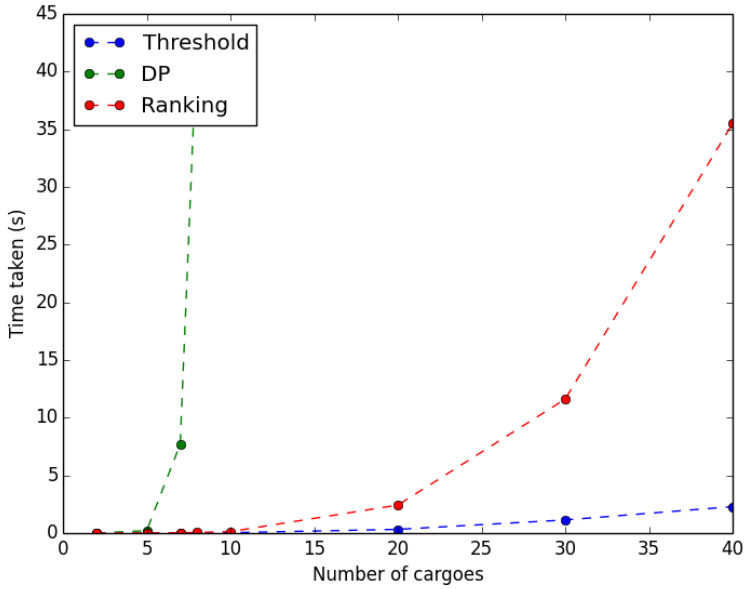


Figure 5-5: Comparison in run-time differing number of cargoes

10 cargoes available at each port. Consistent with the theoretical complexity, the ranking algorithm scales more attractively. It appears to run at an acceptable speed for the number of different cargo options that could practically exist at any given port (up to 40). The threshold heuristic was even faster, and could be scaled further than 40 if needed.

The run time of the 1SL and rollout policies is hard to quantify, as they are online methods that make decisions in real time in the simulation. It was observed that neither added significant overhead onto the run time of the simulation. The rollout heuristic required the threshold policy to be precomputed, so this in an additional computational cost over the myopic policy, although this cost is low considering the threshold policy is extremely fast to calculate.

5.7 Uncertainty in shipping rates

The previous sections focused on the optimal strategy assuming that shipping rates are deterministic. In reality, it could be difficult to forecast shipping rates accurately, even for relatively short time horizons. Therefore, it is necessary to include the

uncertainty involved in future shipping rates in the model.

In this section, we analyze shipping data to form models of price evolution, and aim to capture key insights and dependencies while maintaining simplicity. We use these models to extend the ranking algorithm. Including price uncertainty makes the optimization problem harder for a number of reasons.

- First, it is very difficult to accurately forecast shipping rates, which are a function of a spot market. Like most markets, there is a complex interaction between a number of economic, behavioral and political interactions which drives the rates. Recall that there exist financial instruments known as Forward Freight Agreements (FFAs), which are the rate a shipping company could lock in now for a voyage in the future. If one could accurately predict rates, there would be a lot of money to be made trading these instruments even without optimization.

- Multistage stochastic optimization problems are known as a difficult class of problems to solve. In a MDP framework, including dependencies between time periods causes the state space to explode, because we need to store not only the current price, but also prices in previous periods. If we take a discrete view of prices, the number of different combinations of price over a time period is exponential in nature.

5.7.1 Independent shipping rates

We present a model where shipping rates are independent between routes and time periods. We also assume that rates are discretized. We are able to derive an Independent Rates Ranking Algorithm (IRRA) that is polynomial in the number of cargoes available and the number of discrete prices each cargo can take. Furthermore, this algorithm is optimal under the assumptions of independence and discrete rates.

The independence between rates over time does not imply that the rates have to be stationary, and is indeed most useful when the practitioner has a forecast about how rates will evolve. It is a common practice in the industry to use Forward Freight Agreements (FFA) as a forecast for future shipping rates. The model assumes that the

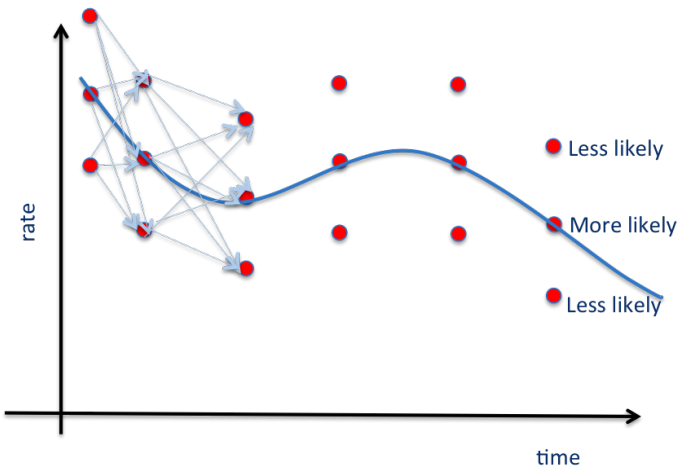


Figure 5-6: Illustration of the possible transitions for the independent discrete errors around a given forecast

shipping rates are independent (discretized) errors around this forecast. This setting is illustrated in Figure 5-6. The values g_{ju} surrounding the FFA can be estimated by looking at the difference between historical rates and FFA as a function of time and discretizing the distribution of the difference. The computational efficiency makes this an attractive algorithm for dealing with uncertainty in rates.

Model

Suppose $\mathcal{G}(i,u,t)$ is a discrete random variable, with M different values and associated probabilities. For notational simplicity, we will drop the (i,t) terms because we will be focusing on calculating the expectation of a given state in this section, and denote this random variable $\mathcal{G}_u$ where the reward is dependent on the cargo chosen u:

$$\mathcal{G}_u = \begin{cases} g_{1u} & \text{w.p } p_{1u} \\ g_{2u} & \text{w.p } p_{2u} \\ \vdots & \vdots \\ g_{Mu} & \text{w.p } p_{Mu} \end{cases}$$

Note that g_{Mu} may equal zero in order to model the case where the cargo u is not available, which would occur with probability p_{Mu}. A naive approach, (similar to the initial model used in Section 5.2) could be to solve this problem by using a state

$(g_1, g_2, ..., g_C)$, which is a vector of observed rates at the time our decision is made. The MDP following from this formulation would have a state space $O(M^C)$, because each cargo could be one of M different prices, making this problem intractable to solve using dynamic programming.

The problem the brokers are trying to solve (from equation (5.3)) at each state, can be easily adapted to the situation with uncertain prices as given in the following equation $V(i,t) = \mathbf{E}[\max_u\{\mathcal{G}(i,u,t) + V(f(i,u))\}]$, or $V = \mathbf{E}[\max_u\{\mathcal{G}_u + V_u\}]$ in reduced notation, where $V(i,t)$ is the expected cost to go from the post decision state (the expected profit from arriving at port i at time t with no knowledge of what cargoes are available). Suppose V_u is known for all u, as would be the case if we had calculated the backwards recursion for all future time periods and states, or if we had an approximate estimate of $V(f(i,u))$. Then V_u is a constant known deterministically.

Define $X_u = \mathcal{G}_u + V_u$ and $W = \max_u\{X_u\}$, where W is the maximal order statistic (see Andrews and Phillips (2005)). Then our problem at each stage reduces to $V = \mathbf{E}[\max_u\{X_u\}] = \mathbf{E}[W]$. Let us denote Ω to be the set of all possible rates we could observe for W. The cardinality is $|\Omega| \leq M_1 + M_2 + ... + M_C$. Let us sort Ω, such that $w_1 = \min_{j,u}\{g_{ju} + V_u\} \leq w_2 \leq .... \leq w_{|\Omega|} = \max_{j,u}\{g_{ju} + V_u\}$. Finally let us define s_k as the cargo index corresponding to the k^{th} ranked observation, and τ_k as the corresponding cargo and and price index uj.

Theorem 8. *If the reward for each cargo is a discrete independent random variable, then the expected reward at each stage is* $V = \sum_{k=1}^{|\Omega|} w_k p_{\tau_k} \prod_{u \in \mathcal{C} \setminus s_k} (\sum_{\{j | g_{ju} + V_u \leq w_{k-1}\}} p_{ju})$.

Proof.

$$
\begin{aligned}
P(W \leq w_k) =& P(X_1 \leq w_k, X_2 \leq w_k ... X_C \leq w_k) && (5.24)\\
=& P(X_1 \leq w_k) P(X_2 \leq w_k),\ .. \ P(X_C \leq w_k) && (5.25)\\
=& \prod_{u \in \mathcal{C}} F_{X_u}(w_k) && (5.26)\\
=& \prod_{u \in \mathcal{C}} (\sum_{\{j | g_{ju} + V_u \leq w_k\}} p_{ju}) && (5.27)
\end{aligned}
$$

The second equality follows from independence of cargoes and rates, and the last

equality follows from the definition of the cumulative density function of a discrete random variable.

$$
\begin{aligned}
P(W = w_k) =& P(W \leq w_k) - P(W \leq w_{k-1}) && (5.28)\\
=& \prod_{u \in \mathcal{C}} F_{X_u}(w_k) - \prod_{u \in \mathcal{C}} F_{X_u}(w_{k-1}) && (5.29)\\
=& \prod_{u \in \mathcal{C}} \Big(\sum_{\{j | g_{ju} + V_u \leq w_k\}} p_{ju} \Big) - \prod_{u \in \mathcal{C}} \Big(\sum_{\{j | g_{ju} + V_u \leq w_{k-1}\}} p_{ju} \Big) && (5.30)\\
=& p_{\tau_k} \prod_{u \in \mathcal{C} \setminus s_k} \Big(\sum_{\{j | g_{ju} + V_u \leq w_{k-1}\}} p_{ju} \Big) && (5.31)\\
& && (5.32)
\end{aligned}
$$

This is the probability that we observe the cargo that can take that price, multiplied by the probability that all other cargoes have an outcome less than that price.

Finally, we can calculate the expectation:

$$
\begin{aligned}
V = \mathbf{E}[W] =& \sum_{w_k \in \Omega} w_k P(W = w_k) && (5.33)\\
=& \sum_{k=1}^{|\Omega|} w_k p_{\tau_k} \prod_{u \in \mathcal{C} \setminus s_k} \Big(\sum_{\{j | g_{ju} + V_u \leq w_{k-1}\}} p_{ju} \Big) && (5.34)
\end{aligned}
$$

□

This algorithm requires approximately $O(TNC^3M^2)$ calculations. At every port and time (TN), $CM \log(CM)$ calculations are required to rank the possible options (cargoes at given price levels), while C^3M^2 calculations are required to calculate the expectation over which cargoes will appear at which price.

This offers a new perspective on the ranking algorithm presented in this chapter. Suppose each X_u has a two point distribution with a probability p_u the cargo is available, in which case there is a reward x_u and a probability $1 - p_u$ it is not available in which case it has a value of 0. This allows us to recover the ranking algorithm from Section 5.3.

Independent rates ranking algorithm (IRRA)

This leads us to propose the independent rates ranking algorithm:

Initialize $V(i,T) = 0,\ \forall i \in \{1,2,...,N\}$. For each time step $t = T, T-1, ..., 1$, for each state $i = 1, ..., N$:

1. Calculate $x_{ju} = g_{ju}(i,t) + V_u(i,t) \qquad \forall u \in \{1,2,...,C\}, j \in \{1,2,...,M\}$

2. We then sort all x_{ju} to get an ordered list $\{w_1, w_2...\}$.

3. Calculate $V(i,t) = \sum_{k=1}^{|\Omega|} w_k p_{\tau_k} \prod_{u \in \mathcal{C} \setminus s_k} (\sum_{\{j | g_{ju} + V_u \leq w_{k-1}\}} p_{ju})$.

We can show, using a similar inductive argument to that used in Section 5.3.2, that this is equivalent to solving the full dynamic program with independent discrete rates.

5.8 Conclusions

In this chapter, we propose a Markov Decision Process formulation of the problem faced by shipping brokers in the spot shipping industry, with limited foresight about the future. This formulation is stochastic and dynamic, and is able to capture the inherent uncertainty in the problem compared to deterministic and static formulations in the literature. We begin with uncertainty on the availability of cargoes, while assuming shipping cargoes follow a deterministic forecast. While a naive dynamic programming implementation is intractable for realistic problem sizes, we introduced and studied a ranking algorithm that is polynomial in the number of cargoes available, and prove that the ranking algorithm is equivalent to solving the DP optimally. We devised several fast approximation methods as benchmarks, including a one step lookahead approach and a rollout approach. We also compared against the average market performance. We constructed a simulation environment that was constructed to facilitate testing a range of policies in an unbiased way using real data from a freight shipping company. This simulation showed that the ranking algorithm outperforms

the other heuristics and the average market performance by 4 -32% over the networks and availability scenarios we tested.

Chapter 6

Conclusions

In this thesis we have studied several key challenges businesses face when trying to adopt prescriptive analytics to make decisions from data. As businesses continue to collect more and more data, and continue to realize the benefits of adopting analytics, this methods will hopefully continue to grow in relevance. Indeed, some early adopters of analytics such as Apple, Alphabet/Google, Amazon, Facebook, Microsoft, GE, and Alibaba Group have established themselves as some of the most valuable companies in the world and continue to have an advantage moving forward.

While this thesis has made an effort to bridge the gap between machine learning, optimization and econometrics, there are many more opportunities for impactful research that remain unaddressed. I also think that each community has a lot to learn from each other as we try and make progress on data-driven optimization. For example, it is very important for the operations research community to be able capture an accurate model between decisions and outcomes, even if it is non-linear and makes the corresponding optimization hard. Unfortunately, in the complex real-word situations, relationships between variables are often non-linear. This is why non-paratmertic machine learning algorithms have been so successful. Although much of the work in this thesis has concerned tree ensembles, I also think there are some very interesting extensions when looking at optimizing over different machine learning functions as objective functions, including neural networks. There has been recent promising work in this area, that can be further extended.

Likewise it is important to accurately capture the treatment effects of the decisions, otherwise optimizing to find the 'optimal' decision may be misleading. The decisions made in the econometrics and treatment effect estimation literature are often very simple, with the majority of the literature dealing with finding average effects for binary treatments. In contrast, the operations research is generally good at making complex combinatorial decisions in a constrained environment. I think combining the work from chapter 2 and chapter 3, to find methods treatment effect estimation for high-dimensional decisions is a very important problem to solve to make help make data-driven optimization more useful and accurate in practice.

Further important problems can be found by combining ideas from machine learning and optimization. In chapter 3 we took a predict then optimize approach where we first estimate a random forest, then optimize over that random forest as an objective function. If the two steps are combined there is potential improved outcomes. One option to do this is to make decisions directly from the data through some specialized loss function. An alternative is train trees with the intent to be used to make decisions rather than predictions. Finally, although I have raised many applications for prescriptive optimization throughout this thesis, I think there are many more opportunities to apply these techniques to different domains. I look forward to seeing the progress that is made in this area. In addition to providing methods and algorithms for solving the general problems raised through this thesis, it is my hope that others will find these issues interesting and contribute to solving them, therefore improving our capability to make data-driven decisions.

Appendix A

Appendix for pricing for heterogeneous products: analytics for ticket reselling

A.1 Derivation of loss function with endogeneity

Proposition 5. *If* $\begin{pmatrix} \epsilon \\ \upsilon \end{pmatrix} \sim N\left(\begin{matrix} 0 \\ 0 \end{matrix}, \begin{pmatrix} \sigma_\epsilon^2 & \rho\sigma_\epsilon\sigma_\upsilon \\ \rho\sigma_\epsilon\sigma_\upsilon & \sigma_\upsilon^2 \end{pmatrix}\right)$ *, and* $u \sim \mathcal{N}(0, \sigma_u^2)$ *is uncorrelated with* ϵ, υ*, then:*

$$\mathbb{E}[Y^*|X] = \sqrt{\sigma_u^2\tau(X)^2\beta_z^2 + \sigma_\epsilon^2 + 2\tau(X)\sigma_\epsilon\sigma_\upsilon + \tau(X)^2\sigma_\upsilon^2} \cdot \Phi^{-1}(\mathbb{E}[Y|X])$$

Proof. We begin by noting that $\mathbb{E}[Z|X] = h(X),\ \mathbb{E}[T|X] = m(X) + \beta_z h(x)$ and

$$\mathbb{E}[Y^*|X] = g(X) + \tau(X)m(X) + \beta_z\tau(X)h(X).$$

Then we have:

$$\mathbb{E}[Y|X] = P\Big(Y^* > 0|X\Big) \tag{A.1}$$

$$= P(g(X) + \tau(X)T + \epsilon > 0|X) \tag{A.2}$$

$$= P(g(X) + \tau(X)(m(X) + \beta_z(h(X) + u) + \upsilon) + \epsilon > 0|X) \tag{A.3}$$

$$= P(\beta_z\tau(X)u + \tau(X)\upsilon + \epsilon > -(g(X) + \tau(X)m(X) + \beta_z\tau(X)h(X))|X) \tag{A.4}$$

$$= P(\beta_z\tau(X)u + \tau(X)\upsilon + \epsilon > -\mathbb{E}[Y^*|X]|X) \tag{A.5}$$

We know that

$$\mathbb{E}[\beta_z\tau(X)u + \tau(X)\upsilon + \epsilon|X] = \beta_z\tau(X)\mathbb{E}[u|X] + \tau(X)\mathbb{E}[\upsilon|X] + \mathbb{E}[\epsilon|X] = 0$$

and

$$Var[\beta_z\tau(X)u + \tau(X)\upsilon + \epsilon|X] \tag{A.6}$$

$$= \beta_z^2\tau(X)^2Var[u|X] + Var[\tau(X)\upsilon + \epsilon|X] + 2Cov(u, \tau(X)\upsilon + \epsilon|X) \tag{A.7}$$

$$= \beta_z^2\tau(X)^2Var[u|X] + \tau(X)^2Var[\upsilon|X] + Var[\epsilon|X] + 2Cov(\upsilon, \epsilon|X) \tag{A.8}$$

$$= \sigma_u^2\tau(X)^2\beta_z^2 + \tau(X)^2\sigma_\upsilon^2 + \sigma_\epsilon^2 + 2\rho\tau(X)\sigma_\epsilon\sigma_\upsilon \tag{A.9}$$

Due to normality, $\beta_z\tau(X)u + \tau(X)\upsilon + \epsilon|X \sim \mathcal{N}(0, \sigma_\epsilon^2 + \tau(X)^2\sigma_\upsilon^2)$. It follows that:

$$\mathbb{E}[Y|X] = \Phi\left(\frac{\mathbb{E}[Y^*|X]}{\sqrt{\sigma_u^2\tau(X)^2\beta_z^2 + \sigma_\epsilon^2 + 2\tau(X)\sigma_\epsilon\sigma_\upsilon + \tau(X)^2\sigma_\upsilon^2}}\right) \tag{A.10}$$

Where Φ is the cdf of standard normal variable. We denote Φ^{-1} the inverse of Φ, then we conclude that

$$\mathbb{E}[Y^*|X] = \sqrt{\sigma_u^2\tau(X)^2\beta_z^2 + \sigma_\epsilon^2 + 2\tau(X)\sigma_\epsilon\sigma_\upsilon + \tau(X)^2\sigma_\upsilon^2} \cdot \Phi^{-1}(\mathbb{E}[Y|X]). \tag{A.11}$$

□

A.2 Proof of theorem 1

Define:

$$\tilde{\Theta} = (\tilde{\boldsymbol{\theta}}, \tilde{\rho}) = \underset{\boldsymbol{\Theta} \in \mathcal{D}_{\Theta}}{\arg\max} \left\{ \frac{1}{n} \sum_{i=1}^{n} \tilde{l}(X_i, T_i, Y_i, Z_i, \boldsymbol{\Theta}) \right\} \tag{A.12}$$

Note that $\tilde{\boldsymbol{\theta}}$ is the maximum likelihood estimator of θ. The proof is separated into three steps:

1. We would verify the sufficient assumptions to utilize the asymptotic theory of maximum likelihood estimators, and thus prove that:

$$\tilde{\boldsymbol{\Theta}} \xrightarrow{p} \boldsymbol{\Theta}$$

2. We would use the uniform convergence assumptions of $\hat{h}$, $\hat{q}$, and $\hat{r}$ along with the maximum likelihood estimator consistency above to prove that:

$$\hat{\boldsymbol{\Theta}} \xrightarrow{p} \boldsymbol{\Theta}$$

3. Using the continuity and compact assumptions, we would show that $\hat{\boldsymbol{\Theta}} \xrightarrow{p} \boldsymbol{\Theta}$ implies:

$$\sup_{X \in M} \|t(\hat{\boldsymbol{\theta}}, X) - t(\boldsymbol{\theta}, X)\| \xrightarrow{p} 0 \qquad \|\hat{\rho} - \rho\| \xrightarrow{p} 0$$

Proof of 1 There are 4 assumptions that once satisfied, are sufficient in order to use the asymptotic theory of maximum likelihood estimators Engle (1994):

(a) The function $l(X, T, Y, Z, \boldsymbol{\Theta})$ is continuous in $\boldsymbol{\Theta}$ for all tuples (X, T, Y, Z).

Proof: As $\tau = t(\boldsymbol{\theta}, X)$ is continuous everywhere in $\boldsymbol{\theta}$ by assumption, the addition, composition of it with continuous functions stay continuous. Thus $l(X, T, Y, Z, \boldsymbol{\Theta})$ is continuous.

(b) There exists an integrable function $D(X, T, Z)$ with respect to $f(X, T, Y, Z|\boldsymbol{\Theta})$

such that:

$$|l(X,T,Y,Z,\Theta)| < D(X,T,Z) \qquad \forall\,(X,T,Y,Z) \in \mathcal{D}_X \times \mathcal{D}_T \times \mathcal{D}_Y \times \mathcal{D}_Z$$

for all τ.

Proof: The domain of $\tau = t(\boldsymbol{\theta}, X)$ is compact (both $\mathcal{D}_\Theta$ and $\mathcal{D}_X$ are compact, so the product space is compact), so there exists $(\boldsymbol{\theta}^*, X^*)$ such that:

$$|t(\boldsymbol{\theta}^*, X^*)| \geq |t(\boldsymbol{\theta}, X)| \qquad \forall\,(\boldsymbol{\theta}, X) \in D_\Theta \times \mathcal{D}_X$$

Then we have:

$$\tilde{f}(T,X,Z) = \Phi^{-1}(r(X))\sqrt{1 + 2|\rho\sigma_v|t(\boldsymbol{\theta}^*, X^*) + ({\sigma_v}^2 + \sigma_u^2\beta^2)t(\boldsymbol{\theta}^*, X^*)^2}$$
$$+ |t(\boldsymbol{\theta}^*, X^*)||T - q(x)| + \frac{\rho}{\sigma_v} v \geq |f(T,X,Z,\Theta)|$$

for all $\boldsymbol{\theta} \in D_\Theta$.

As log is strictly increasing, we then have:

$$D(X,T,Z) = |\log(\Phi(-\tilde{f}(T,X,Z)))| > |l(X,T,Y,Z,\boldsymbol{\Theta})|$$

For all $\boldsymbol{\theta} \in D_\Theta$.

(c) The domain of $\boldsymbol{\Theta}$ is compact.

Proof: This follows directly from the assumption that $\boldsymbol{\theta} \in D_\Theta$, and D_Θ is compact.

(d) $\boldsymbol{\Theta} \neq \boldsymbol{\Theta}_0 \quad \Leftrightarrow \quad l(\cdot, \boldsymbol{\Theta}) \neq l(\cdot, \boldsymbol{\Theta}_0)$.

Proof: The reverse direction is trivial. For the forward direction, we would show that there exists (X,T,Y) such that $\boldsymbol{\Theta} \neq \boldsymbol{\Theta}_0 \quad \Rightarrow \quad l(X,T,Y,Z,\boldsymbol{\Theta}) \neq l(X,T,Y,Z,\boldsymbol{\Theta}_0)$.

We first note that if $\boldsymbol{\theta} = \boldsymbol{\theta}_0$ and $\rho \neq \rho_0$, then we have $f(X,T,Y,Z,\boldsymbol{\theta},\rho) \neq$

$f(X,T,Y,Z,\boldsymbol{\theta},\rho_0)$ if we take $v>0$, then $\frac{\partial f(\tau,\rho,X,T,Z,v)}{\partial \rho}>0$.

Therefore, we would focus on the case that $\theta \neq \theta_0$. We first take $Y=1$, and thus the log-likelihood function becomes:

$$l(X,T,1,Z,\boldsymbol{\theta}) = \log(\Phi(f(\boldsymbol{\theta},X,T,Z,\rho)))$$

As log and Φ are injective functions, we only need to show that there exists (X,T,Z) such that $(\boldsymbol{\theta},\rho) \neq (\boldsymbol{\theta}_0,\rho_0) \quad \Rightarrow \quad f(\boldsymbol{\theta},X,T,Z,\rho) \neq f(\boldsymbol{\theta}_0,X,T,Z,\rho_0)$.

Let $\boldsymbol{\theta},\boldsymbol{\theta}_0 \in \mathcal{D}_\Theta$ and $\boldsymbol{\theta} \neq \boldsymbol{\theta}_0$. By identifiability of τ, there exists $X_i \in \mathcal{D}_X$ such that $\tau(\boldsymbol{\theta},X_i) \neq \tau(\boldsymbol{\theta}_0,X_i)$. From our model, as $T = m(X)+\beta Z+v$ where v is normally distributed, $\mathcal{D}_T = (-\infty,\infty)$. Thus, take T_i sufficiently large enough so that the following is true:

$$\begin{aligned}\frac{\partial f(\tau,X_i,T_i,Z_i,\rho)}{\partial \tau} &= \frac{\rho\sigma_v + (\sigma_v^2+\sigma_u^2\beta^2)\tau(X)}{k(\tau,\rho,X)}\Phi^{-1}(\mathbb{E}[Y \mid X]) + T_i - \mathbb{E}[T|X] \\ &> |\rho-\rho_0| \max_{k\in(\rho,\rho_0)} |\frac{\partial f(\tau,X_i,T_i,Z_i,\rho)}{\partial \rho}|_{\rho=k}|\end{aligned}$$

for all τ. Then f is monotonic in τ with such (X_i,T_i,Z_i) for all ρ and thus we have $f(\boldsymbol{\theta},X_i,T_i,Z_i,\rho) \neq f(\boldsymbol{\theta}_0,X_i,T_i,Z_i,\rho_0)$ as required.

Thus we have:

$$\tilde{\boldsymbol{\Theta}} \xrightarrow{p} \boldsymbol{\Theta}$$

Furthermore, according to the asymptotic normality of MLE estimators, we have:

$$\sqrt{n}(\tilde{\boldsymbol{\Theta}} - \boldsymbol{\Theta}) \xrightarrow{p} N(\mathbf{0},\boldsymbol{\Sigma})$$

For some covariance matrix $\boldsymbol{\Sigma}$.

Proof of 2 As $\mathcal{D}_X, \mathcal{D}_\Theta$ compact, we have $|\tau| \leq c < \infty$ for some c. Moreover, since $\mathcal{D}_X$ is compact, and $0 < r(X) < 1$, let the maximum and minimum value of $r(X)$ be $b<1$ and $a>0$ respectively. Then we note that $\Phi^{-1}(r(X))$ is bounded.

Using the uniform convergence assumptions of $\hat{q}(X)$, $\hat{r}(X)$, ρ, σ_u, σ_v, β, v, then

by Slutsky's lemma we have:

$$\sup_{(X_i,T_i,Y_i,Z_i)\in\mathcal{D}_X\times\mathcal{D}_T\times\mathcal{D}_Y\times\mathcal{D}_Z} |\hat{l}(X,T,Y,Z,\Theta) - \tilde{l}(X,T,Y,Z,\Theta)| \to 0$$

The supremum is valid as all of the domains are compact. As $\mathbf{\Theta}$ uniquely maximizes the likelihood, we thus have

$$|\tilde{\mathbf{\Theta}} - \hat{\mathbf{\Theta}}| \xrightarrow{p} 0$$

Therefore, we have:

$$\hat{\mathbf{\Theta}} \xrightarrow{p} \mathbf{\Theta}$$

If we have $\sup_{X\in\mathcal{D}_x} O(n^{1/2+\delta})|\hat{q}(X) - q(X)| \xrightarrow{p} 0$, $\sup_{X\in\mathcal{D}_x} O(n^{1/2+\delta})|\hat{r}(X) - r(X)| \xrightarrow{p} 0$, and similarly for all nuisance parameters for some $\delta > 0$, then, we have:

$$O(n^{1/2+\delta})|\tilde{\mathbf{\Theta}} - \hat{\mathbf{\Theta}}| \xrightarrow{p} 0$$

Thus, we have:

$$\sqrt{n}(\hat{\mathbf{\Theta}} - \mathbf{\Theta}) \xrightarrow{p} 0$$

Using the asymptotic normality of MLE above.

Proof of 3 Then, as $\tau : \mathcal{D}_\Theta \times \mathcal{D}_X$ is continuous in $\boldsymbol{\theta}$, we have that:

$$\|t(\hat{\boldsymbol{\theta}}, X) - t(\boldsymbol{\theta}, X)\| \xrightarrow{p} 0 \quad \forall X \in M$$

Now as τ has a compact domain (products of compact spaces are compact), pointwise convergence implies uniform convergence:

$$\sup_{X\in M} \|t(\hat{\boldsymbol{\theta}}, X) - t(\boldsymbol{\theta}, X)\| \xrightarrow{p} 0$$

As τ is continuous in X with probability 1. By consistency of our estimator Θ

as above, we also have:

$$\hat{\rho} \xrightarrow{p} \rho$$

With the additional assumptions, we can have that:

$$\sqrt{n}\|t(\hat{\boldsymbol{\theta}}, X) - t(\boldsymbol{\theta}, X)\| \xrightarrow{p} 0 \quad \forall X \in M$$

And thus:

$$\sup_{X \in M} \sqrt{n}\|t(\hat{\boldsymbol{\theta}}, X) - t(\boldsymbol{\theta}, X)\| \xrightarrow{p} 0$$

Similarly for $\hat{\rho}$. $\square$

A.3 Proof of optimal price strategy

Proof. For notational simplicity, we will prove the optimal price strategy theorem when $\rho = 0$, and $\sigma_v = 1$. The proof for the more general case follows naturally. We have for a given ticket, the optimal price T^* is the following:

$$T^* = \text{argmax}_T \ \hat{R}^*(X, T) = T \cdot \Phi\Big(\sqrt{1 + \hat{\tau}(X)^2} \cdot \Phi^{-1}(\mathbb{E}[Y|X]) + \hat{\tau}(X)(T - m(X))\Big) \tag{A.13}$$

1. If $\hat{\tau}(X) \geq 0$, then since $\Phi(\cdot)$, the standard normal cdf, is a increasing function, by composition, the objective function $\hat{R}^*(X, \cdot)$ is increasing in price T, and the reseller should choose the highest price possible got the ticket.

2. If $\hat{\tau}(X) < 0$, we calculate the first and second derivatives of $\hat{R}^*(X, \cdot)$ with respect to price T. We have:

$$\frac{\partial \hat{R}^*}{\partial T}(X, T) = \Phi\Big(\sqrt{1 + \hat{\tau}(X)^2} \cdot \Phi^{-1}(\mathbb{E}[Y|X]) + \hat{\tau}(X)(T - m(X))\Big) \tag{A.14}$$

$$+ T \cdot \hat{\tau}(X) \cdot \Phi'\Big(\sqrt{1 + \hat{\tau}(X)^2} \cdot \Phi^{-1}(\mathbb{E}[Y|X]) + \hat{\tau}(X)(T - m(X))\Big) \tag{A.15}$$

and

$$\frac{\partial^2 \hat{R}^*}{\partial^2 T}(X,T) = 2\hat{\tau}(X)\cdot \Phi^{'}\Big(\sqrt{1+\hat{\tau}(X)^2}\cdot\Phi^{-1}(\mathbb{E}[Y|X]) + \hat{\tau}(X)(T-m(X))\Big) \tag{A.16}$$

$$+ T\cdot\hat{\tau}(X)^2\cdot\Phi^{''}\Big(\sqrt{1+\hat{\tau}(X)^2}\cdot\Phi^{-1}(\mathbb{E}[Y|X]) + \hat{\tau}(X)(T-m(X))\Big) \tag{A.17}$$

Since for all $u\in\mathbb{R}\ :\ \Phi^{''}(u) = -u\cdot\Phi^{'}(u)$, we can rewrite the last equation as :

$$\frac{\partial^2 \hat{R}^*}{\partial^2 T}(X,T) = \Big(2\hat{\tau}(X) - T\cdot\hat{\tau}(X)^2\cdot\Big(\sqrt{1+\hat{\tau}(X)^2}\cdot\Phi^{-1}(\mathbb{E}[Y|X]) + \hat{\tau}(X)(T-m(X))\Big)\Big)\cdot g(T,X) \tag{A.18}$$

where $g(T,X) = \Phi^{'}\Big(\sqrt{1+\hat{\tau}(X)^2}\cdot\Phi^{-1}(\mathbb{E}[Y|X]) + \hat{\tau}(X)(T-m(X))\Big) > 0$ forall T.

The sign of $\frac{\partial^2 \hat{R}^*}{\partial^2 T}$ depends on the sign of :

$$\Big(2\hat{\tau}(X) - T\cdot\hat{\tau}(X)^2\cdot\Big(\sqrt{1+\hat{\tau}(X)^2}\cdot\Phi^{-1}(\mathbb{E}[Y|X]) + \hat{\tau}(X)(T-m(X))\Big)\Big) =$$
$$\hat{\tau}(X)\Big(-\hat{\tau}(X)^2\cdot T^2 - \hat{\tau}(X)\cdot(\sqrt{1+\hat{\tau}(X)^2}\cdot\Phi^{-1}(\mathbb{E}[Y|X]) - \hat{\tau}(X)m(x))\cdot T + 2\Big) \tag{A.19}$$

which is a quadratic function in T. We can then show that there exist $\underline{\mathrm{T}} < \bar{T}$ such that $\frac{\partial^2 \hat{R}^*}{\partial^2 T}(X,T) > 0$ for $T < \underline{\mathrm{T}}$, $\frac{\partial^2 \hat{R}^*}{\partial^2 T}(X,T) \leq 0$ for $T\in[\underline{\mathrm{T}},\bar{T})$ and $\frac{\partial^2 \hat{R}^*}{\partial^2 T}(X,T) > 0$ for $T > \bar{T}$, and we characterize $\underline{\mathrm{T}}$ and $\bar{T}$ as the following:

$$\begin{aligned}
\underline{\mathrm{T}} &= \frac{\sqrt{1+\hat{\tau}(X)^2}\cdot\Phi^{-1}(\mathbb{E}[Y|X]) - \hat{\tau}(X)m(x)}{-2\hat{\tau}(X)} \\
&\quad - \frac{\sqrt{(\sqrt{1+\hat{\tau}(X)^2}\cdot\Phi^{-1}(\mathbb{E}[Y|X]) - \hat{\tau}(X)m(x))^2 + 8}}{-2\hat{\tau}(X)}, \\
\bar{T} &= \frac{\sqrt{1+\hat{\tau}(X)^2}\cdot\Phi^{-1}(\mathbb{E}[Y|X]) - \hat{\tau}(X)m(x)}{-2\hat{\tau}(X)}
\end{aligned} \tag{A.20}$$

$$+\frac{\sqrt{(\sqrt{1+\hat{\tau}(X)^2}\cdot\Phi^{-1}(\mathbb{E}[Y|X])-\hat{\tau}(X)m(x))^2+8}}{-2\hat{\tau}(X)} \tag{A.21}$$

One can show easily that $\underline{\mathrm{T}} < 0$ and $\bar{T} > 0$. We can also show that $\frac{\partial^2 \hat{R}^*}{\partial^2 T}(X, 0) > 0$ and $\frac{\partial^2 \hat{R}^*}{\partial^2 T}(X, \bar{T}) < 0$. We conclude that $\frac{\partial \hat{R}^*}{\partial T}(X, T) = 0$ has a unique solution T^* and we have $\frac{\partial \hat{R}^*}{\partial T}(X, T) > 0$ for $T < T^*$ and $\frac{\partial \hat{R}^*}{\partial T}(X, T) \geq 0$ for $T \geq T^*$, which makes the revenue function unimodal $\hat{R}^*$. In addition, we have $T^* \leq \bar{T}$.

□

A.4 Feature importance and few feature formulation

A.4.1 Feature importance

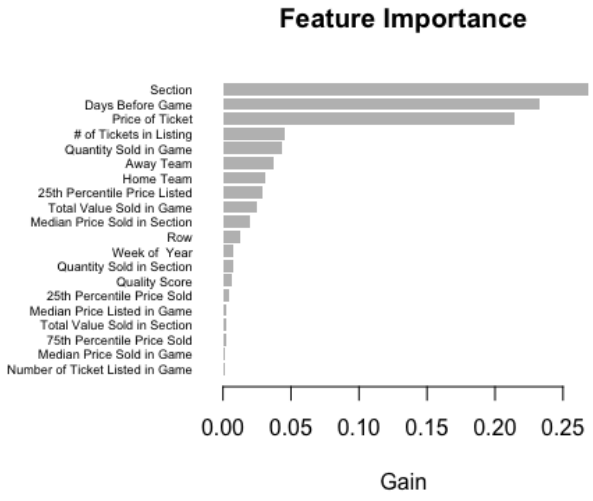


Figure A-1: Feature importance as evaluated by boosted trees

Figure A-1 shows the relative feature importance as evaluated by a `lightgbm` gradient boosted trees predictor. As expected, price is a very important factor for determining the probability of a ticket selling. The section of the ticket, corresponding to how close the seat is to the court, is also very important, as is the number of days until the game. When there is only a few days until the game, the ticket has a

lower probability of selling. Previous sales and prices, and the teams playing are also important predictors.

A.4.2 Few feature formulation

Given the high feature importance of some features compared to others, we would test a model with few features as a baseline for comparison. The specific features chosen are:

- Section
- Days Before Game
- Price of Ticket

This corresponds to the three features with the highest variable importance as illustrated in A-1. We then present the predictive accuracy results under this case:

Algorithm	Misclassification error	AUC
Two stage LGBM	0.181	0.832
One stage LGBM	0.187	0.830
probit iv	0.217	0.745
LGBM iv	0.197	0.820

Table A.1: Predictive accuracy on NBA data iv

We see that the top 3 features, although overwhelmingly important by variable importance metric, is insufficient for a well-performing model, as even the best-performing model under this restriction is less accurate than the worst model under the full set of features. This suggests that the additional list of features are essential for an accurate model.

A.5 Correlation between tickets

In this section, we investigate the validity of our assumption that the selling behavior between different tickets listed at the same time are not highly correlated, and thus the decision to use a single-item optimization model is justified.

In a market with many players (thousands in the case of a ticket exchange), each seller is a price taker and will not have an significant effect on the market. Apart from capturing the (assumed) exogenous market effects, a seller may be concerned about cross effects between tickets they are selling, and whether they need to solve a joint pricing problem. To explore this, we look at pairs of tickets which are listed at the same time and try to see if the covariance is significant.

The sample covariance of two tickets are as below:

$$\hat{\text{Cov}}(Y_i, Y_j) = \hat{\mathbb{P}}[Y_i = 1, Y_j = 1] - \hat{\mathbb{P}}[Y_i = 1]\hat{\mathbb{P}}[Y_j = 1] \tag{A.22}$$

We would focus on the mean of the sample covariance among tickets. That is, for a set of tickets M, we would investigate:

$$\hat{\mu} = \frac{1}{|\{i,j\} \mid i,j \in M \ \&\ \text{Colist}(i,j) = 1|} \sum_{i,j \in M : \text{Colist}(i,j)=1} \hat{\text{Cov}}(Y_i, Y_j) \tag{A.23}$$

Here $\text{Colist}(i,j)$ is the binary function such that $\text{Colist}(i,j) = 1$ if ticket i and ticket j has a minimum of one day in overlap in listing time, and that ticket i and ticket j belong to the same game. We only consider ticket pairs in the same game, as inter-game correlation is expected to be low due to the nature of the events, while most of the correlation (if there is any) should be intra-game.

If the correlation between tickets are small, we expect $\hat{\mu} \approx 0$. Thus we would utilize the Two One-Sided Equivalence Test (TOST):

$$H_0 : |\hat{\mu}| > \Delta \qquad H_1 : |\hat{\mu}| < \Delta$$

And we intend to find the minimum value of Δ such that H_0 can be rejected on the

5% level. We would utilize two separate probit regressions to estimate $\mathbb{P}[Y_i = 1]$ and $\mathbb{P}[Y_j = 1]$, and use the residuals to determine the variance of $\hat{\mu}$. under strong law of large numbers, $\hat{\mu}$ is approximately normal, and thus we would utilize a t-test to test this hypothesis. The details of the calculations on the variance of $\hat{\mu}$ is contained in A.5.1.

We trained the two probit regressions on historical data and evaluated it against 4.47 Million pairs of tickets, formed through a random set of tickets with size $|M|=100,000$. The first probit model (n=4,470,000) is used to estimate $\hat{\mathbb{P}}[Y_i = 1, Y_j = 1]$, while the second probit model (n=100,000) estimates $[Y_i = 1]$.

The minimum Δ is:

$$\Delta_m = 0.0110$$

Thus, we can reject the hypothesis that $|\hat{\mu}|> 0.01$ on the 5% level. Since $\Delta_m \approx 0$, this provides statistical evidence that the ticket market, on average, is uncorrelated, and thus we can utilize single-ticket optimization.

A.5.1 Technical details for covariance test

For the covariance term:

$$\hat{\text{Cov}}(Y_i, Y_j) = \hat{\mathbb{P}}[Y_i = 1, Y_j = 1] - \hat{\mathbb{P}}[Y_i = 1]\hat{\mathbb{P}}[Y_j = 1] \tag{A.24}$$

We assume that our estimates approximately follows the normal distribution with iid samples:

$$\hat{\mathbb{P}}[Y_i = 1, Y_j = 1] \dot{\sim} N(\mathbb{P}[Y_i = 1, Y_j = 1], \sigma_1^2) \qquad \hat{\mathbb{P}}[Y_i = 1] \dot{\sim} N(\mathbb{P}[Y_i = 1], \sigma_2^2) \quad \forall\, i, j \tag{A.25}$$

Then we have that:

Theorem 9. *The bias and the variance of the covariance is:*

$$\mathbb{E}[\hat{\text{Cov}}(Y_i, Y_j)] = \text{Cov}(Y_i, Y_j)$$

$$\mathbb{V}[\hat{\text{Cov}}(Y_i, Y_j)] = \sigma_1^2 + \sigma_2^4$$

The proof of unbiasedness is straightforward and is be omitted. For the variance, we note the following lemma:

Lemma 7. *For independent normal variables $X, Y \sim N(0, \sigma^2)$, we have that:*

$$XY = c(Q - R) \tag{A.26}$$

Where $c = \frac{1}{2}\sigma^2$, and $Q, R \sim \chi_1^2$ are independent.

Then using the lemma, we have:

$$\begin{aligned}
\mathbb{V}[\hat{\text{Cov}}(Y_i, Y_j)] &= \mathbb{V}\left[\hat{\mathbb{P}}[Y_i = 1, Y_j = 1]\right] + \mathbb{V}\left[\hat{\mathbb{P}}[Y_i = 1]\hat{\mathbb{P}}[Y_j = 1]\right] \\
&= \sigma_1^2 + c^2(\mathbb{V}[Q] + \mathbb{V}[R]) \\
&= \sigma_1^2 + \sigma_2^4 \quad \square
\end{aligned}$$

Where $Q, R \sim \chi_1^2$ and $c = \frac{1}{2}\sigma_2^2$. Thus, under the assumption that there is no correlation ($\text{Cov}(Y_i, Y_j) = 0$), we derive the distribution of μ as:

$$\mathbb{E}[\hat{\mu}] = 0 \qquad \mathbb{V}[\hat{\mu}] = \frac{\sigma_1^2 + \sigma_2^4}{|\{i, j\} \mid i, j \in M \;\&\; \text{Colist}(i, j) = 1|} \tag{A.27}$$

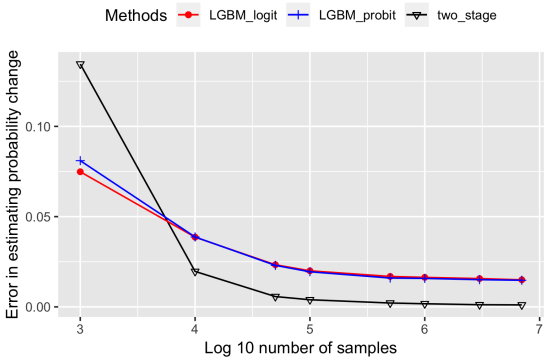
Then the variances σ_1^2 and σ_2^2 are replaced with their estimates $\hat{\sigma}_1^2$ and $\hat{\sigma}_1^2$ from the models for the single probabilities $\mathbb{P}[Y_i = 1]$ and cross-probabilities $\mathbb{P}[Y_i = 1, Y_j = 1]$ respectively. In our simulation, the estimated numbers were $\sigma_2^2 = 0.1098$ and $\sigma_1^2 = 0.2033$ respectively.

A.6 Additional numerical experiments

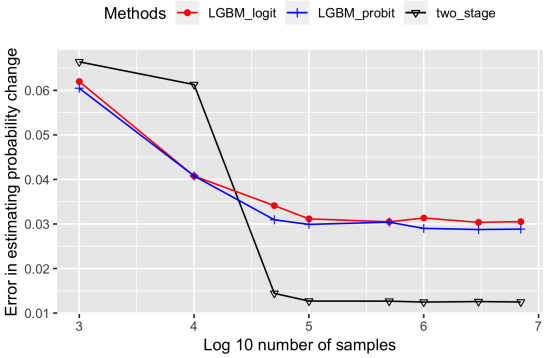
A.6.1 Treatment effect error with sample size

We also explore the effect of sample size on the ability to predict the change in probability associated with a fixed treatment change of 0.5 in Figure A-2. We observe

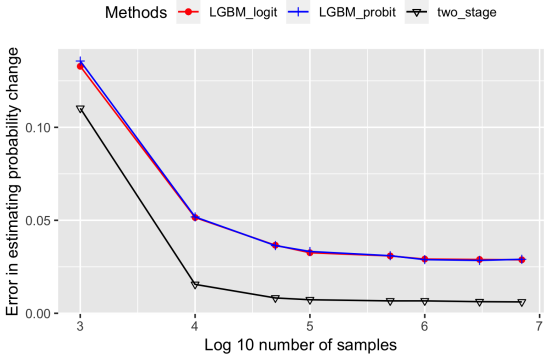
that the two stage approach is generally able to converge to high accuracy estimates. In dataset 2 and 4, we observe that the two stage approach may require more samples than a single stage, since it requires high accuracy estimates of $\mathbb{E}[T|X]$ and $\mathbb{E}[Y|X]$ which are also affected by a small sample size.



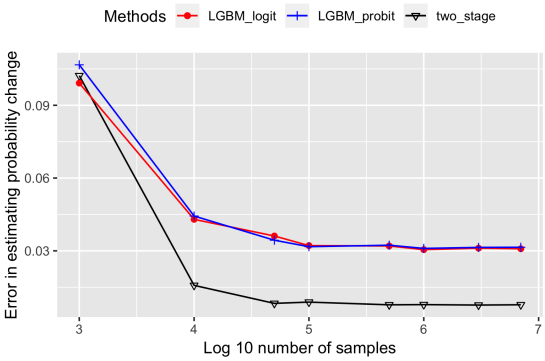
(a) Dataset 2: simple variable effect treatment



(b) Dataset 3: linear relationships



(c) Dataset 4: non-linear $g(x)$, linear $m(x), \tau(x)$



(d) Dataset 5: non-linear relationships

Figure A-2: Estimating treatment effect for changing sample size

A.6.2 Optimal treatment prescription simulations

To explore the prescriptive element of our algorithm, we can calculate the optimal treatment for an adaptation of our synthetic data to a revenue maximization setting. Using notation from section **??**, we find the optimal price $T^* = \arg\max_T T \cdot P(Y = 1|X, T)$, where $P(Y = 1|X, T)$ is calculated analytically for the respective synthetic datasets. We can compare against the estimated optimal treatment $\hat{T} = \arg\max_T T \cdot$

$\hat{f}(X,T)$, where $\hat{f}(X,T)$ is the estimate of $P(Y=1|X,T)$ for each of our models. In the case of the two stage approach, this is equivalent to solving (2.17). We compare the optimality gap in revenue for each estimated optimal treatment. We use the same experimental set up as above and use an exhaustive line search to find optimal T in each optimization problem:

dataset	two stage	one stage probit	one stage logit
constant treatment	0	0	0
linear relationships	**0.0017**	0.0082	0.0082
non-linear relationships	**0.0028**	0.0104	0.0134

Table A.2: Difference between revenue at prescriptive price $\hat{T}$ and true optimal price T^*

We can observe that in each of our synthetic datasets, the improvement in estimation of the treatment effect translates to an improvement in selecting the optimal treatment. In the next section we outline how we adapted this approach to StubHub's data.

A.7 Backtesting framework

To understand the performance of our algorithm on real-life data without immediately deploying it to production, we need to conduct backtesting experiments to see how different models perform over periods in a trading environment. To do so, we generated some simplifying assumptions that allowed us to construct a backtesting framework for testing the performance of the models on past years' data efficiently. The backtesting framework uses two models; the testing model which is used to buy/price tickets and a baseline model which is used to evaluate whether the tickets will sell. The backtesting procedure is as follows:

1. We enter all the parameters needed to execute the backtesting framework, including:

- **Model Retrain Period** - How frequent do we want to retrain the model, if we want to do so?
- **Baseline Retrain Period** - How frequent do we want to retrain the baseline, if we want to do so?
- **Test Model Type** - The algorithm used to train the model for predictive decisions.
- **Baseline Model Type** - The algorithm used to train the baseline for counterfactuals.
- **Time Model Type** - The algorithm used to estimate the time needed for a sold ticket to be actually sold.
- **Buffer Type** - What type of buffer do we want to have for our optimization to be comfortable to buy a ticket? A percentage or an absolute gain, or both?
- **Maximum Percentage Increase Allowed on Tickets** - What is the upper limit of our optimization?
- **Minimum Percentage Gain Needed to buy Tickets** - A minimum percentage gain needed (of its optimal value over current price) for the optimization to decide to buy the ticket, if the appropriate buffer type is chosen.
- **Minimum Absolute Gain Needed to buy Tickets** - A minimum absolute gain needed (of its optimal value over current price) for the optimization to decide to buy the ticket, if the appropriate buffer type is chosen.

2. The specified testing model is trained on 1,000,000 ticket samples in the immediate past of the starting date. The specified baseline model is trained on all ticket samples contained in the entire dataset so it serves as an oracle.

3. At the start of the day (00:00), we assume all the tickets that are listed on that day are available.

4. We execute the testing model to evaluate all the tickets listed on the market and buy tickets according to optimization procedure in section §2.5.3.

5. After the ticket is bought, the testing model immediately uses the calculated optimal price to relist the ticket either dynamically or in a static way. For the dynamic setting, we discretize the selling period decisions to the following cutoffs:

$$(100, 90, 80, 70, 60, 50, 45, 40, 35, 30, 29, 28, \cdots, 1, 0)$$

   We consider selling prices discretized by 5 dollar intervals centered around the original price, with a minimum of 50% below the listing price and a maximum of 50% over the listing price. The selling probabilities for each period is decided through evaluating the baseline model at the endpoints of these periods, while accounting for market state changes. While evaluating, the daysBefore feature is set to the number of days on the end points, and for each market feature M_i, we define it using the following evolution equation:

$$M_i(t) = M_i(0) + \mu_i t + \sigma_i t\epsilon$$

   Where μ_i and σ_i are parameters fitted through historical data, and $M_i(0)$ indicates the market state feature at time of listing. Epsilon is a standard $(0, 1)$ Normally distributed variable. Therefore, each market feature is assumed to be a Gaussian random walk. We run 10 random walks each for each ticket to calculate the selling probability matrix. The results are averaged.

6. The testing model tries to sell everything in its inventory at the calculated optimal price, and the baseline model evaluates the selling. We evaluate the selling decision based on 1 random walk for each ticket. The evaluation selling probabilities are based on the baseline model.

7. At the end of the day (23:59), we assume all the tickets that are removed on that day are removed, and all tickets that perish that day are also removed.

8. The day count moves forward by one day, and we return to Step 3. If a retraining period for the testing model or the baseline model is specified, and the period is due on that day, we return to Step 2.

Appendix B

Appendix for optimizing objective functions determined from random forests

B.1 Benders subproblem results

B.1.1 Calculation of optimal dual solution

In this subsection, we find an explicit analytical expression of a viable solution to the Benders dual subproblem (3.4), as is done in Mišić (2017) (although our form is different due to different problem and formulation). We will introduce some notations, and for what follows we will abstract from using the subscript t for tree t. Denote p^* the leaf that the primal solution (x, g) belongs to and let $\mathcal{N}(p^*) = \{i \in N \mid \text{i is a parent of } p^*\}$, i.e., the set of parent nodes of leaf p^*. We denote $\mathcal{RP}(p^*)$ (and respectively $\mathcal{LP}(p^*)$) the set of parent nodes for which p^* is a right descendant (respectively the set of parent nodes for which p^* is a left descendant). For a interior node $i \in N$, let $\mathcal{L}^r(i)$ (respectively $\mathcal{L}^l(i)$) be the set of leafs that are on the right path (respectively left path) of node i. Let g be the primal solution associated with leaf p^*. We have the following proposition to characterize the optimal solution to (3.4).

Proposition 6 (Optimal solution to (3.4))**.** *An optimal solution* $(\mathbf{r_i}, \mathbf{l_i}, \mathbf{v}, \mathbf{u_i})_{\mathbf{i}\in\mathbf{N}}$ *to (3.4) is characterized as the following:* $v = S_{p*}$ *and for* $i \in N$*:*

$$r_i = \max_{j\in\mathcal{L}^r(i)} (S_j - S_{p*})^+ \quad \textit{if } i \in \mathcal{LP}(p^*),\ 0 \textit{ otherwise.}$$

$$l_i = \max_{j\in\mathcal{L}^l(i)} (S_j - S_{p*})^+ \quad \textit{if } i \in \mathcal{RP}(p^*),\ 0 \textit{ otherwise.}$$

$$u_i = \min_{j\in\mathcal{L}^l(i)\cup\mathcal{L}^r(i)} (S_{p*} - S_j) \textit{ if } i \notin \mathcal{N}(p^*),\ 0 \textit{ otherwise.}$$

Proof. An optimal solution to the dual subproblem (3.4) is a feasible solution that achieves -by strong duality- an optimal objective equal to the primal subproblem's optimal objective. For a specific vector (x, g), this is equal to the score S_{p^*} of leaf p^* that x belongs to. Using the above notation, notice that for $i \in \mathcal{LP}(p^*)$: $g_i = 0$ and for $i \in \mathcal{RP}(p^*)$: $g_i = 1$. Now, we verify that the proposed solution achieves the optimal dual objective:

$$\sum_{i\in N}(r_i g_i + l_i(1-g_i)) + v - u_0 = \sum_{i\in\mathcal{LP}(p^*)} r_i g_i + \sum_{i\in\mathcal{RP}(p^*)} l_i(1-g_i)) + S_{p^*} - 0 = S_{p^*}.$$

Next, we verify that the proposed solution is feasible. First note that by construction that $r_i \geq 0$ and $l_i \geq 0$ for $i \in N$. For a right leaf $j \in \mathcal{L}_r$, if $p_j \in \mathcal{N}(p^*)$, unless $j = p^*$ (in which case $p_j \in \mathcal{RP}(p^*)$ and $r_{p_j} - u_{p_j} + v = 0 - 0 + S_{p^*} = S_{p^*}$), p_j has to be in $\mathcal{LP}(p^*)$ and thus $r_{p_j} - u_{p_j} + v = (S_j - S_{p^*})^+ - 0 + S_{p^*} = S_j$. Otherwise, if $p_j \notin \mathcal{N}(p^*)$, we have $r_{p_j} - u_{p_j} + v = 0 - \min_{k\in\mathcal{L}^l(i)\cup\mathcal{L}^r(i)}(S_{p*} - S_k) + S_{p^*} \geq 0 - (S_{p*} - S_j) + S_{p*} = S_j$. Likewise, we can show that for any left leaf $j \in \mathcal{L}_l$, the constraint $l_{p_j} - u_{p_j} + v \geq S_j$ is verified.

For a right interior node $i \in N_r$, if both i and p_i are in $\mathcal{N}(p^*)$, then $u_i = u_{p_i} = 0$ and the constraint $r_{p_i} - u_{p_i} + u_i \geq 0$ is satisfied. If neither i nor p_i are in $\mathcal{N}(p^*)$, then notice that, since for any interior node i, $\mathcal{L}^r(i) \subseteq \mathcal{L}^l(p_i) \cup \mathcal{L}^l(p_i)$ and $\mathcal{L}^r(i) \subseteq \mathcal{L}^r(p_i) \cup \mathcal{L}^l(p_i)$, then $u_{p_i} \leq u_i$, and the constraint $r_{p_i} - u_{p_i} + u_i \geq 0$ is again satisfied. Else, if $i \notin \mathcal{N}(p^*)$, and $p_i \in \mathcal{N}(p^*)$, then p_i has to be in $\mathcal{LP}(p^*)$ because i is a right node. We check then that $r_{p_i} - u_{p_i} + u_i = \max_{j\in\mathcal{L}^r(p_i)}(S_j - S_{p*})^+ - 0 + \min_{j\in\mathcal{L}^l(i)\cup\mathcal{L}^r(i)}(S_{p*} - S_j) = 0$, so the

constraint is satisfied. We can show likewise that for any left interior node $i \in N_l$, $l_{p_j} - u_{p_j} + u_i \geq 0$.

We conclude that the proposed solution, being feasible and achieving the optimal objective is indeed an optimal solution to the dual problem (3.4). □

B.1.2 Benders Pareto-optimal cuts

Next, we show in detail how we generate Pareto-optimal cuts in our particular setting where the subproblem is non-linear with x (but still linear with respect to the auxiliary variable vector g), to improve the Benders formulation. Note that these cuts are disaggregated in trees (for each tree there is a subproblem). Hence to simplify notations, we will omit the t subscript from all parameters. That is to say we will look at r_i instead of r_i^t and likewise g_i, l_i, v, u_0, γ. Our Benders cuts have a dependency only upon the $g = (g_i)_i$ variables, but in what follows, we will use the notation $g(x)$ to refer to the auxiliary variable g associated with x. We write the cuts in a general form $\gamma \leq \gamma^m(g(x))$ where $\gamma^m(g(x)) = \sum_{i \in N}((r_i^m - l_i^m)g_i(x) + l_i^m) + v^m - u_0^m$.

Definition 7. *A Benders cut $\gamma \leq \gamma^m(g(x))$ dominates another Benders cut $\gamma \leq \gamma^s(g(x))$ if $\gamma^m(g(x)) \leq \gamma^s(g(x))$ for all feasible x and is a strict inequality for at least one feasible x .*

The contrapositive is that if there exists a feasible solution x^* such that $\gamma^m(g(x^*)) > \gamma^s(g(x^*))$, then $\gamma \leq \gamma^m(g(x))$ does not dominate $\gamma \leq \gamma^s(g(x))$.

Definition 8. *A Benders cut $\gamma \leq \gamma^m(g(x))$ is considered Pareto-optimal if it is not dominated by any other cuts.*

This means, if for any other Benders cut $\gamma \leq \gamma^s(g(x))$ there exists a feasible solution x^* such that $\gamma^m(g(x^*)) < \gamma^s(g(x^*))$, then $\gamma \leq \gamma^m(g(x))$ is Pareto-optimal. Let $\mathcal{G} = \text{conv}(\{g(x), x \in \mathcal{X}\})$, where $\text{conv}(S)$ is the convex hull of S and let g^0 a point in the relative interior of $\mathcal{G}$. While Magnanti and Wong (1981) show a method to generate Pareto optimal cuts when the subproblem dual's objective is linear with

the primal variable vector x, we generalize the method to non-linear dependency in x. This gives rise to the following proposition:

Proposition 7. *Let $\mathcal{D}$ be the feasible set for the subproblem dual. Given an incumbent solution $\bar{x}$ of the master problem, let $V(\bar{x})$ be the optimal objective of the primal subproblem. If :*

$$\begin{aligned}(r_i^0, l_i^0, v^0, u_i^0) = \underset{r_i,l_i,v,u_i}{argmin} & \sum_{i\in N}((r_i - l_i)g_i^0 + l_i) + v - u_0 \\ subject\ to\ & \sum_{i\in N}((r_i - l_i)g_i(\bar{x}) + l_i) + v - u_0 \leq V(\bar{x}) \qquad (D^0(\bar{x})) \\ & (r_i, l_i, v, u_i) \in \mathcal{D}\end{aligned}$$

then the Benders cut $\gamma \leq \gamma^0(g(x)) = \sum_{i\in N}((r_i^0 - l_i^0)g_i(x) + l_i^0) + v^0 - u_0^0$ is Pareto optimal.

Proof. Suppose $\gamma \leq \sum_{i\in N}((r_i^0 - l_i^0)g_i(x) + l_i^0) + v^0 - u_0^0$ is not Pareto optimal, that is there is a cut $\gamma^s(g(x)) = \sum_{i\in N}((r_i^s - l_i^s)g_i(x) + l_i^s) + v^s - u_0^s$ with $(r_i^s, l_i^s, v^s, u_i^s) \in \mathcal{D}$ that dominates $\gamma^0(g(x))$. We then have :

$$\gamma^s(g(x)) \leq \gamma^0(g(x)) \quad \forall x \in \mathcal{X} \tag{B.1}$$

For any point $g \in \mathcal{G}$, there exist some finite number of points $x_1, x_2, \cdots, x_j \in \mathcal{X}$ such that g can be expressed as $\sum_{j=1}^{l} \lambda_j g(x_j)$, $\sum_{j=1}^{l} \lambda_j = 1$, $\lambda_j \geq 0$. Then from (B.1):

$$\gamma^s(g) \leq \gamma^0(g) \quad \forall g \in G \tag{B.2}$$

Using inequality (B.1), we get for $x = \bar{x}$ that $\gamma^s(g(\bar{x})) \leq \gamma^0(g(\bar{x})) \leq V(\bar{x})$, so $(r_i^s, l_i^s, v^s, u_i^s)$ must be an optimal solution to $(D^0(\bar{x}))$. Then from inequality (B.2) and the definition of $(r_i^0, l_i^0, v^0, u_i^0)$ we have:

$$\gamma^s(g^0) = \gamma^0(g^0) \tag{B.3}$$

Since $\gamma^s(g(x))$ dominates $\gamma^0(g(x))$:

$$\gamma^s(g(\hat{x})) < \gamma^0(g(\hat{x})) \tag{B.4}$$

for some $\hat{x} \in \mathcal{X}$. In addition, because $g^0 \in \mathcal{G}^0$, there exists (see Rockafellar (2015)) a scalar $\theta > 1$ such that $z = \theta g^0 + (1-\theta)g(\hat{x}) \in \mathcal{G}$. We multiply (B.3) by θ and inequality (B.4) by $(1-\theta)$, which is negative and reverses the inequality, then add them both which gives:

$$\gamma^0(g(z)) < \gamma^s(g(z)) \tag{B.5}$$

which leads to the desired contradiction. □ □

Let us comment on the result and its value. The problem $D^0(\bar{x})$ is a very similar LP to the original dual subproblem $D(x)$. It can then be solved efficiently using standard software. We will need to recover $V(\bar{x})$ the optimal objective of the primal subproblem (through a simple tree search). Note also that our result is stated for an arbitrary point g^0 in the relative interior of $\mathcal{G}$, and consequently we only need to find such a point *once*. LP techniques such as in (Schrijver 2000, Cartis et al 2006) can find a point in the relative interior of a polyhedron efficiently.

B.2 Hoeffding difference proof

Lemma 8. *Let $Z_i = (X_i, W_i)$ be iid random variables each bounded by the interval $[a, b]$, with $E[X_i] = \mu \leq E[W_i] = \tau$.*

$$P(\sum_{i=1}^{n} \frac{X^i}{n} \geq \sum_{i=1}^{n} \frac{W_i}{n}) \leq e^{\frac{-n}{2(a-b)^2}(\mu-\tau)^2}$$

Proof. We have

$$\begin{aligned} P(\sum_{i=1}^{n}\frac{X_i}{n} \geq \sum_{i=1}^{n}\frac{W_i}{n}) &= P(\sum_{i=1}^{n}\frac{X_i}{n} - \sum_{i=1}^{n}\frac{W_i}{n} + n(\mu-\tau) \geq n(\mu-\tau)) \\ &\leq e^{\frac{-2n^2(\mu-\tau)^2}{n(2b-2a)^2}} \qquad \textit{(Hoeffding's Inequality)} \\ &\leq e^{\frac{-n}{2(a-b)^2}(\mu-\tau)^2} \quad \square \end{aligned} \tag{B.6}$$

□

B.3 Bound of suboptimality in the size of M

Note: Let $(S_i)_{i=1,\cdots,M}$ be iid random variables with a cumulative distribution function F. Denote $\mathcal{M}(\epsilon) = \{i | S_{max} - S_i \geq \epsilon\}$ and $\mathcal{M}(\epsilon)^C = \{i | S_{max} - S_i < \epsilon\}$. Then, for any $\epsilon > 0$, $E[\frac{|\mathcal{M}(\epsilon)|}{|\mathcal{M}(\epsilon/2)^C|+1}] \quad c]M \to \infty = \quad \mathcal{O}(1)$.

We have $\frac{|\mathcal{M}(\epsilon)|}{|\mathcal{M}(\epsilon/2)^C|+1} \leq \frac{M}{|\mathcal{M}(\epsilon/2)^C|+1}$ and $E[\frac{|\mathcal{M}(\epsilon)|}{|\mathcal{M}(\epsilon/2)^C|+1}] \leq M \cdot E[\frac{1}{|\mathcal{M}(\epsilon/2)^C|+1}]$.

Note that for any $s \in [a,b]$, conditioned on the event $S_{max} = s$, $|\mathcal{M}(\epsilon/2)^C|$ is a binomial random variable $B(M-1, p_\epsilon^s)$ with $p_\epsilon^s = P(S_1 > s - \epsilon/2) = 1 - F(s-\epsilon/2)$. By definition of the expectation, if we denote $X \sim B(N,p)$, then : $\mathbb{E}(\frac{1}{1+X}) = \sum_{k=0}^{N} \frac{1}{1+k} \cdot \binom{N}{k} p^k (1-p)^{N-k} = \frac{1}{(N+1)p} \cdot \sum_{k=0}^{N} \binom{N+1}{k+1} \cdot p^{k+1}(1-p)^{N-k} = \frac{1}{(N+1)p} \cdot (1-(1-p)^{N+1})$. Hence, it follows that:

$$\begin{aligned} E[\frac{1}{|\mathcal{M}(\epsilon/2)^C|+1}] &= \int_a^b E[\frac{1}{|\mathcal{M}(\epsilon/2)^C|+1} | S_{max} = s] P(S_{max} = s) \\ &= \int_a^b \frac{1}{M \cdot p_\epsilon^s} \cdot (1 - (1-p_\epsilon^s)^M) \cdot M \cdot f(s) F(s)^{M-1} ds \\ &= \int_a^b \frac{1}{(1 - F(s-\epsilon/2))} \cdot \underbrace{(1 - F(s-\epsilon/2)^M)}_{\leq 1} \cdot f(s) F(s)^{M-1} ds \end{aligned} \tag{B.7}$$

Because F is non-decreasing, we do know that $0 < 1 - F(b-\epsilon/2) \leq 1 - F(s-\epsilon/2) \leq$

1 for $s \in [a, b]$, then it results from (B.7) that:

$$M \cdot E[\frac{1}{|\mathcal{M}(\epsilon/2)^C|+1}] \leq \frac{1}{(1-F(b-\epsilon/2))} \cdot \underbrace{\int_a^b M \cdot f(s)F(s)^{M-1}ds}_{=1} = \frac{1}{(1-F(b-\epsilon/2))}. \tag{B.8}$$

We conclude that $E[\frac{|\mathcal{M}(\epsilon)|}{|\mathcal{M}(\epsilon/2)^C|+1}] \quad \underset{[}{c}]M \to \infty = \quad \mathcal{O}(1)$.

B.4 Hazard rate dominance numerical experiments

In proof 4, it is necessary to assume that Y_i have hazard rate dominance when ordered according to their score. In this section, we provide numerical evidence that suggests that the hazard rate dominance assumption is reasonable for Y_i for large n.

In the following experiment, we study the distribution of the score for a random forest evaluated at a finite number of sample points $\{x_1, ..., x_s\}$. We use the Juror dataset described in section 3.8.2 to train a large random forest, and the sample points $\{x_1, ..., x_s\}$ are features of juries from a testing set with the labels removed. For each x_i we can calculate the score on a set $\mathcal{K}_n$ of n randomly selected trees, and average to get the random forest estimate $Y(x_i) = \sum_{k \in \mathcal{K}_n} \frac{X^k(x_i)}{n}$. For each trial, we randomly sample the set of trees $\mathcal{K}_n$ to get a distribution of the score $Y(x_i)$. We are interested in exploring the hazard function $\frac{f_{Y(x_i)}(y)}{1-F_{Y(x_i)}(y)}$, and particular seeing if there is a hazard rate ordering $w_1, ..., w_s$ such that $\frac{f_{Y(x_{w_i})}(y)}{1-F_{Y(x_{w_i})}(y)} \geq \frac{f_{Y(x_{w_{i+1}})}(y)}{1-F_{Y(x_{w_{i+1}})}(y)}$ $\forall y, i \in 1, ..., s$.

To estimate the pdf $f_{Y(x_i)}(y)$ we used the `Scikit-learn KernelDensity` estimation package (Pedregosa et al. (2011a)) with a gaussian kernel and bandwidth of 0.4. To estimate $F_{Y(x_i)}(y)$ we use the empirical CDF function `StatsModels ECDF` (Seabold and Perktold (2010)). Each of the four figures shows the hazard rate of $Y(x_i)$ at the different sample points for $n = 10, 30, 100, 300$. On the y axis is the log hazard rate, and the x axis is the domain of the random variable $Y(x_i)$. Each line corresponds to the hazard rate of a sample point $Y(x_i)$. We observe that as n gets large the $Y(x_i)$ do appear to form an ordering such that $Y(x_{w_{i+1}}) \geq_{HRD} Y(x_{w_{i+1}})$,

since the hazard rate lines do not appear to cross over each other. This means that the hazard rate to the right dominates the hazard rate to the left. In comparison, for n small, we see the lines do cross over so hazard rate dominance does not hold. Intuitively, this is because $Y(x_i)$ converges almost surely to $S(x_i)$ according to the strong law of large numbers, while $Y(x_i)\sqrt{(n)}$ converges to a normal distribution by the CLT, for which hazard rate dominance holds assuming the standard deviation is not too large.

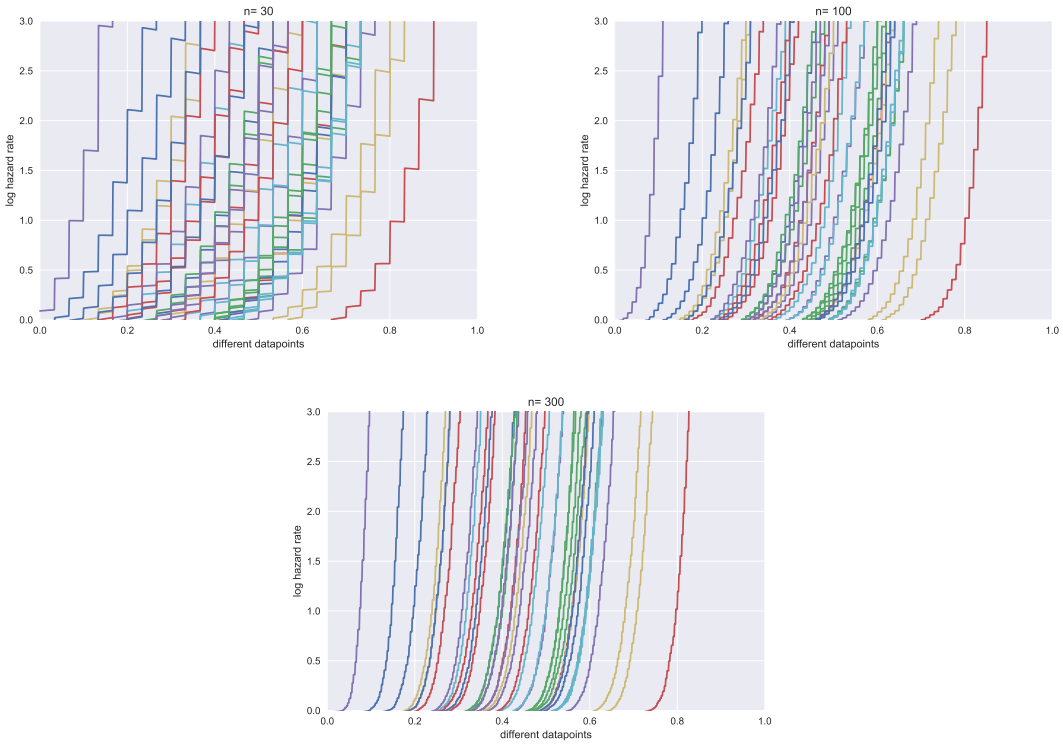


Figure B-1: Hazard rate dominance for Y_i as the number of trees sampled (n) grows

B.5 Proof of bounds for cross validation

Proposition 8. *Let x^{CV} be the solution derived from the cross validation for optimization procedure, over τ forests $R_1, ..., R_\tau$. Then*

$$\frac{1}{\tau}\sum_{i=1}^{\tau} R_i(x^{CV}) \leq z^* \leq \frac{1}{\tau}\sum_{i=1}^{\tau} \max_{x\in\mathcal{X}} R_i(x)$$

Proof. The lower bound follows from the fact that x^{CV} is a feasible solution for the global problem of optimizing over entire forest. Also if $x^* = \arg\max_{x\in\mathcal{X}} \sum_{i=1}^{\tau} R_i(x)$, then $R_i(x^*) \leq \max_{x\in\mathcal{X}} R_i(x)$ so $\sum_{i=1}^{\tau} R_i(x^*) \leq \sum_{i=1}^{\tau} \max_{x\in\mathcal{X}} R_i(x)$, proving the upper bound. □ □

Corollary 3. *For all subsets* $\{j = 1, ..., \tau\}$, *trees* $\{t = 1, ..., n\}$, *and partitions of the target forest* $\{i = 1, .., M\}$, *let* $\{X_{ij}^t\}_{i=1,\cdots,M;j=1,\cdots,\tau}^{t=1,\cdots,T}$ *be independent random variable bounded by the interval* $[a, b]$, *with* $E[X_{ij}^t] = S_i$. *Let* $Y_{ij} = \sum_{t=1}^{n} \frac{X_{ij}^t}{n}$ *and* $i_j^* = \{\arg\max_i Y_{ij} | \omega_i \cap \mathcal{X} \neq \emptyset\}$, *while* $S_{max} = \{\max_i S_i | \omega_i \cap \mathcal{X} \neq \emptyset\}$, *and* $i^{CV} = \arg\min S_{max} - S_{i_j^*}$. *Define* $\mathcal{M}(\epsilon) = \{i | S_{max} - S_i > \epsilon\}$. *Then for any* $\epsilon > 0$, $P(S_{max} - S_{i^{CV}} > \epsilon) \leq (|\mathcal{M}|(\epsilon) e^{\frac{-n}{2(b-a)^2}\epsilon^2})^{\tau}$

$$\begin{aligned} P(S_{max} - S_{i^{CV}} > \epsilon) &= P(\cup_{k=\{1,..,\tau\}} S_{max} - S_{i_k^*} > \epsilon) \\ &= P(S_{max} - S_{i^*} > \epsilon)^{\tau} \\ &\leq (|\mathcal{M}| e^{\frac{-n}{2(b-a)^2}\epsilon^2})^{\tau} \end{aligned}$$

B.6 Optimizing over a logistic regression objective

Due to the sigmoid function $\sigma(\cdot)$ being an increasing function, maximizing $\sigma(x)$ given $x \in \mathcal{X}$ is equivalent to maximizing x. Therefore, the optimization problem we solve is:

$$\begin{aligned} \max \quad & \sum_i \beta_i x_i \\ \text{s.t} \quad & x \in \mathcal{X} \end{aligned}$$

Note that β_i are the coefficients of the logistic regression trained on the training data and $\mathcal{X}$ is the feasible set of the optimization problem. The linear regression also has this form, except the coefficients are those of the trained linear regression.

B.7 Cross-validation

In tables B.1 and B.2, we show further experimental results from section 3.9.3, solved over a larger variety of trees.

# trees	20 ($N_{nodes} = 1187$)					40 ($N_{nodes} = 2312$)					60 ($N_{nodes} = 3319$)				
# subsets	20	10	5	2	1	20	10	5	2	1	20	10	5	2	1
# trees per subset	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3
% optimality	95.00%	85.00%	85.00%	55.00%	35.00%	97.50%	87.50%	60.00%	72.50%	72.50%	95.00%	88.33%	58.33%	58.33%	66.67%
% time taken	10.87%	5.37%	2.72%	1.03%	0.38%	10.71%	5.53%	2.90%	1.35%	0.83%	9.58%	4.92%	2.24%	0.84%	0.45%
# trees	80 ($N_{nodes} = 4537$)					120 ($N_{nodes} = 6499$)					160 ($N_{nodes} = 8592$)				
# subsets	20	10	5	2	1	20	10	5	2	1	20	10	5	2	1
# trees per subset	4	4	4	4	4	6	6	6	6	6	8	8	8	8	8
% optimality	96.25%	96.25%	91.25%	66.25%	66.25%	93.33%	93.33%	75.83%	73.33%	73.33%	95.63%	95.63%	95.63%	84.38%	84.38%
% time taken	5.55%	3.12%	1.74%	0.92%	0.67%	9.50%	5.11%	2.75%	0.96%	0.47%	9.08%	5.14%	2.40%	1.03%	0.56%
# trees	300 ($N_{nodes} = 16238$)					500 ($N_{nodes} = 27401$)					700 ($N_{nodes} = 37624$)				
# subsets	20	10	5	2	1	20	10	5	2	1	20	10	5	2	1
# trees per subset	15	15	15	15	15	25	25	25	25	25	35	35	35	35	35
% optimality	98.67%	98.67%	96.67%	81.67%	76.67%	98.80%	97.80%	95.60%	92.80%	87.20%	98.00%	95.57%	95.57%	97.14%	93.43%
% time taken	12.51%	7.18%	5.12%	3.32%	2.70%	14.50%	6.68%	3.76%	1.48%	0.78%	15.18%	8.47%	3.67%	1.70%	1.05%

Table B.1: Cross-validation performance on jury dataset

# trees	20 ($N_{nodes} = 33126$)				30 ($N_{nodes} = 48884$)				40 ($N_{nodes} = 66848$)			
# subsets	10	5	2	1	10	5	2	1	10	5	2	1
# trees per subset	2	2	2	2	3	3	3	3	4	4	4	4
% optimality	94.2%	89.2%	88.5%	70.5%	98.8%	86.8%	82.4%	93.4%	98.5%	96.9%	83.6%	92.8%
% time taken	161.7%	78.1%	30.2%	15.2%	122.3%	65.7%	24.5%	11.1%	89.0%	45.1%	17.4%	8.7%
# trees	50 ($N_{nodes} = 79662$)				60 ($N_{nodes} = 96792$)				70 ($N_{nodes} = 113232$)			
# subsets	10	5	2	1	10	5	2	1	10	5	2	1
# trees per subset	5	5	5	5	6	6	6	6	7	7	7	7
% optimality	99.1%	92.7%	92.7%	92.7%	100.0%	99.1%	93.9%	93.9%	99.2%	99.2%	95.2%	95.2%
% time taken	72.0%	36.3%	14.0%	7.4%	74.0%	35.4%	13.6%	6.7%	82.2%	51.1%	42.0%	52.6%

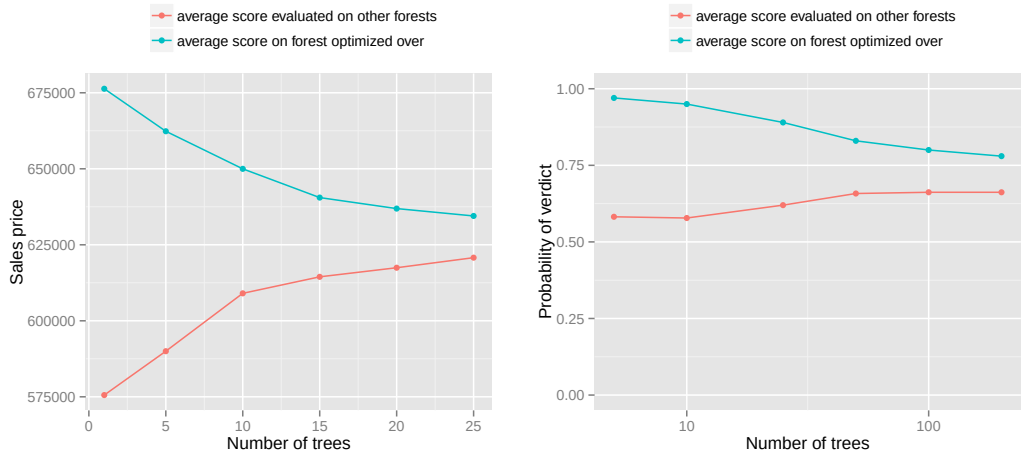
Table B.2: Cross validation performance on housing dataset

B.7.1 Cross-validation, tree size and overfitting

To better understand the cross validation algorithm, we can analyze the *in-forest* performance compared to the *out-of-forest* performance for a particular solution, as shown in figures B-2a and B-2b. This allows us to explore the behavior of the bounds discussed in section 3.6.1 and explore sensitivity relative to the number of trees optimized over.

In the following experiments, 10 random forests were trained from the data, and the best solution selected for each by optimizing over a random forest objective function. We then tested in and out-of-forest accuracy of the solutions, where *out-of-forest* accuracy is the average score predicted for a solution evaluated on a different random

forest to the one it was selected. The blue line shows the prediction for the solution on the forest it was optimized over (*in-forest* score) and is the upper bound on the optimal solution referred to in proposition 8. We see strong evidence of overfitting for forests with a small number of trees, but as the size of the random forest we optimize over is increased, the effects of overfitting are reduced. This is indicated by the blue line approaching the red line, which shows the average score on forests that the solution was not optimized for (*out-of-forest* score). Clearly, the *out-of-forest* performance improves as the number of as trees in the forest increases, because the solution is more general and less overfit to a particular subset of trees. These results are similar to comparing the in-sample prediction accuracy to the OOB accuracy when training a random forest for prediction. Likewise, this gap tends to decrease with the number of trees in the forest due to less overfitting.



(a) House price in-forest compared to out-of-forest

(b) Jury score in-forest compared to out-of-forest

Figure B-2: In forest vs out of forest behavior

Table B.3: Standard deviation out of forest price for housing dataset

# of trees in each subset	1	5	10	15	20	25
Stdev out of forest score	83309	27097	26885	19156	16207	16293

This difference between *in-forest* performance compared to the *out-of-forest* performance helps understand why cross-validation is effective. If the difference is large,

the *in-forest* score is not necessarily a reliable performance metric due to overfitting, but through cross-validation we can evaluate how the solution performs *out-of-forest*.

Furthermore, by comparing figure B-2 and table B.3 we also observe that there tends to be higher variance in the out-of-forest score when the difference between *in-forest* score and *out-of-forest* score is large. Because of the high variance of the solutions, it is still likely that some solutions are high quality, and cross-validation enables us to identify and use these solutions.

B.8 Jury behavior

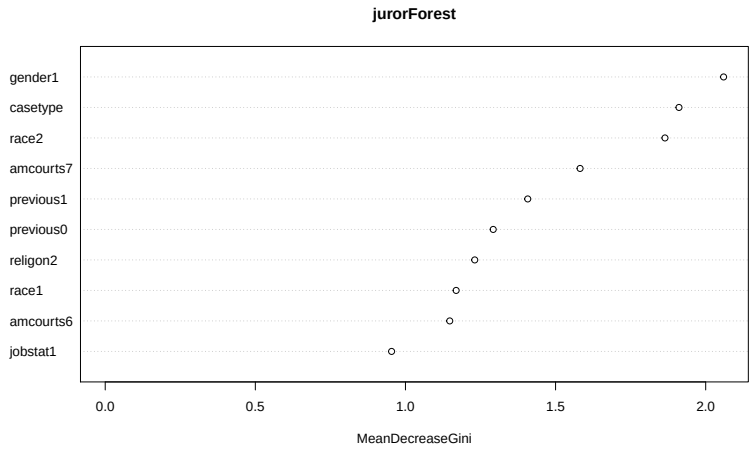


Figure B-3: Feature importance random forest

Figure B-3 shows the importance of the features in the random forest. This suggests that the gender balance (gender1) of the jury is the most important, followed by the case type. Unfortunately case type is a feature of the case not of the jury, so it is not something we can optimize over. The number of people of white people (race2) in the jury is apparently important, as is the number of jurors who say they "place a great deal of trust in the justice system" (amcourts7). The number of people on the jury who had done jury service (previous1, previous0) is also of some importance, as is the number of jurors who identified as religious (religion2).

Appendix C

Appendix for a ranking algorithm for tramp shipping in the spot market

C.1 Data and inference

We will outline how we used this data to calibrate the model in this section. Note that the following parameters have been used in our models so far:

- p_u - the probability that cargo u is available
- $R(l(u), d(u), t)$ - the reward associated with taking cargo u at time t
- $C(i, l(u), t)$ - the ballast cost associated with taking traveling from port i to pick up cargo u at time t
- $D(i, u)$ - time taken to discharge cargo u from port i
- T - time horizon

We assume that ballast cost $C(i, l(u), t)$ is constant per day, as the costs of operating a vessel are fuel and crew expenses are fixed. For the simulations in this section, we assume the reward of taking a cargo $R(l(u), d(u), t)$ is calculated based on a snapshot of the world shipping rates. This is necessary to maintain interpretability of the results, as rationale behind why different decisions are made becomes very difficult

when there are highly variable shipping rates. We explore models with non-stationary and uncertain shipping rates in section 5.7. We had access to the historical movements of all Aframax vessels (a specific class of oil tanker) over the two years from $2013-2015$. There are roughly $35,000$ voyages over this period. From the time taken for each voyage $D(i,u)$, we can estimate the distance between each pair of ports by taking a simple average over each time they have been traversed.

Our assumption of independence of cargo availability between time periods is explored in figure C.1, which shows the autocorrelation between time lags for different regions cargoes might be present. Given the low correlation coefficients across different regions, we believe independence is a reasonable assumption to make to enable tractability. This allows us to calculate the probability of picking a cargo up in a region as the fraction of the days a cargo is available in a region over the time horizon.

C.2 Dynamic programming example

Let us look at a simple example to illustrate how this algorithm works, illustrated by figure C-1. Suppose we have a 3 port network with ports $\{1,2,3\}$. We are a ship currently located at port 1 at time t. Suppose that there are two possible cargoes, one that takes us to port 2 with reward $G(1,2,t)$ and takes a time $D(1,2)$, and one that takes us to port 3 with reward $G(1,3,t)$ and takes a time $D(1,3)$. If both of these are available, the availability vector is $\mathcal{S}=(-,1,1)$ which happens with probability p_2p_3. To decide which cargo to take we do the following calculation to compare the cost of taking cargo 2, cargo 3 or waiting

$$\begin{aligned} J(1,t,(-,1,1)) = \max\{&G(1,2,t)+\mathbf{E}_{\mathcal{S}}J(2,t+D(1,2),\mathcal{S}),\ ... \\ &...G(1,3,t)+\mathbf{E}_{\mathcal{S}}J(3,t+D(1,3),\mathcal{S}),\mathbf{E}_{\mathcal{S}}J(1,t+1,\mathcal{S})\} \end{aligned} \tag{C.1}$$

Notice that this requires the calculation of the residual reward $\mathbf{E}_{\mathcal{S}}J(2,t+D(1,2),\mathcal{S})$, $\mathbf{E}_{\mathcal{S}}J(3,t+D(1,3),\mathcal{S})$ if we were to choose each of these options, which is where the majority of the computational effort is required. Let's calculate $\mathbf{E}_{\mathcal{S}}J(1,t,\mathcal{S})$ to show

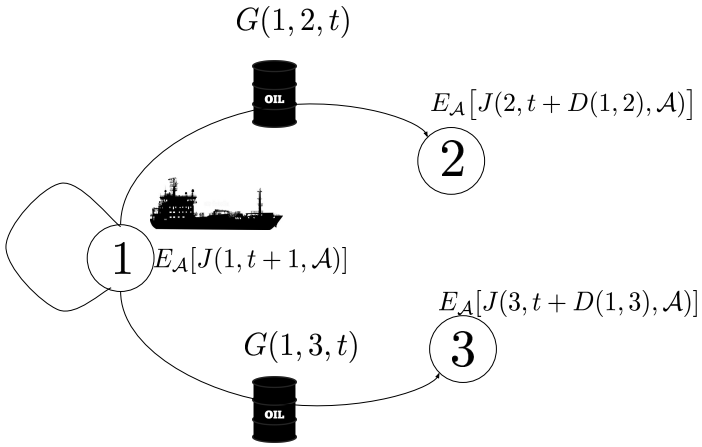


Figure C-1: example of how to solve problem with at port 1 both cargoes available

Time Lag	1	2	3	4	5	6	7	8	9	10	11
Arabian Gulf	1	0.049	-0.024	0.014	0.051	0.026	0.039	0.087	0.124	0.063	-0.010
Arctic Ocean Barents Sea	1	-0.005	-0.005	-0.005	-0.005	-0.005	-0.005	-0.005	-0.005	-0.005	-0.005
Argentina Uruguay.	1	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
Bahamas	1	0.024	-0.001	-0.001	0.024	-0.001	0.050	-0.027	-0.001	-0.001	0.076
Baltic Sea Low	1	-0.005	-0.005	-0.005	-0.005	-0.005	-0.005	-0.005	-0.005	-0.005	-0.005
Baltic Sea Upper	1	0.017	-0.002	0.073	0.045	0.008	0.017	0.045	0.007	0.007	-0.002
Black.Sea	1	0.023	-0.001	0.064	0.035	-0.013	0.051	0.010	0.070	0.020	0.020
Brazil	1	-0.004	-0.004	-0.004	-0.004	-0.004	-0.004	0.163	-0.004	-0.004	-0.004
Caribs North Coast South America	1	-0.055	0.011	-0.042	0.007	-0.002	-0.008	-0.027	0.022	-0.034	0.026
Caribs North Coast South America	1	-0.009	0.075	-0.009	-0.009	-0.009	-0.009	-0.009	0.075	-0.009	-0.009
Continent	1	0.030	-0.016	-0.016	0.030	-0.016	-0.016	-0.016	-0.016	0.030	0.030
East Coast Canada	1	-0.022	-0.003	0.016	-0.003	0.016	0.016	-0.003	-0.022	-0.003	0.016
East Coast Central America	1	-0.007	-0.007	-0.007	-0.007	-0.007	-0.007	-0.007	-0.007	-0.007	-0.007
East Coast Mexico	1	0.056	0.002	-0.024	0.029	0.005	-0.009	0.023	0.014	-0.011	-0.035
Eastern Mediterranean	1	0.031	0.023	0.013	0.072	0.048	0.043	0.039	0.055	0.071	0.042
Murmansk	1	-0.001	-0.001	-0.001	-0.001	-0.001	-0.001	-0.001	-0.001	-0.001	-0.001
North Sea	1	-0.011	-0.011	-0.010	-0.010	-0.010	0.057	-0.010	-0.010	-0.010	-0.010
North West Africa	1	-0.004	-0.004	-0.004	-0.004	-0.004	-0.004	-0.004	-0.004	-0.004	-0.004
Pakistan West Coast India	1	-0.034	0.005	0.044	0.005	0.024	-0.015	0.044	0.083	0.005	0.005
Red Sea	1	-0.021	0.016	0.052	0.016	-0.021	-0.020	0.016	-0.020	-0.020	-0.020
Sea of Marmara	1	-0.006	-0.006	-0.006	-0.006	-0.006	-0.006	-0.006	-0.006	-0.006	-0.006
United Kingdom	1	0.330	0.163	-0.004	-0.004	-0.004	-0.004	-0.004	-0.004	-0.004	-0.004
US Atlantic Coast	1	-0.009	-0.009	-0.009	-0.009	-0.009	-0.009	-0.009	-0.009	-0.009	0.075
US Gulf	1	-0.008	-0.008	0.049	-0.023	0.044	0.023	0.016	0.022	0.019	0.019
West Africa	1	0.035	-0.026	-0.016	-0.040	-0.002	-0.045	0.002	0.016	-0.007	0.007
Western Mediterranean	1	-0.022	0.071	0.047	0.059	0.069	0.057	0.051	0.099	0.051	0.055

Table C.1: Autocorrelation for different loading areas over multiple time lags

how this works.

There are 4 different combinations of cargo availability for this example $\mathcal{S} =\in \{(-,0,0),(-,1,0),(-,0,1),(-,1,1)\}$ (either no cargoes are available, only cargo 2 is available, only cargo 2 is available,both cargoes are available), with probabilities $(1-p_2)(1-p_3), p_2(1-p_3), (1-p_2)p_3$ and p_2, p_3 respectively, so we calculate the expected cost to go under each of these additional scenarios.

$$\begin{aligned}
J(1,t,(-,0,0)) =& \mathbf{E}_{\mathcal{S}} J(1,t+1,\mathcal{S}) \\
J(1,t,(-,1,0)) =& \max\{G(1,2,t) + \mathbf{E}_{\mathcal{S}} J(2,t+D(1,2),\mathcal{S}), \mathbf{E}_{\mathcal{S}} J(1,t+1,\mathcal{S})\} \\
J(1,t,(-,0,1)) =& \max\{G(1,3,t) + \mathbf{E}_{\mathcal{S}} J(3,t+D(1,3),\mathcal{S}), \mathbf{E}_{\mathcal{S}} J(1,t+1,\mathcal{S})\} \\
\mathbf{E}_{\mathcal{S}} J(1,t,\mathcal{S}) =& (1-p_2)(1-p_3) J(1,t,(-,0,0)) + p_2(1-p_3) J(1,t,(-,1,0)) \\
& \ldots + (1-p_2)p_3 J(1,t,(-,0,1)) + p_2 p_3 J(1,t,(-,1,1))
\end{aligned} \tag{C.2}$$

Suppose cargo 3 is the most profitable over the remaining time horizon, followed by cargo 2 and finally waiting so that $G(1,3,t) + \mathbf{E}_{\mathcal{S}} J(3,t+D(1,3),\mathcal{S}) > G(1,2,t) + \mathbf{E}_{\mathcal{S}} J(3,t+D(1,3),\mathcal{S}) > \mathbf{E}_{\mathcal{S}} J(1,t+1,\mathcal{S})$. Then we have that

$$\begin{aligned}
\mathbf{E}_{\mathcal{S}} J(1,t,\mathcal{S}) =& (1-p_2)(1-p_3)\mathbf{E}_{\mathcal{S}} J(1,t+1,\mathcal{S}) \ldots \\
& \ldots + p_2(1-p_3)(G(1,2,t) + \mathbf{E}_{\mathcal{S}} J(2,t+D(1,2),\mathcal{S})) \ldots \\
& \ldots + p_3(G(1,3,t) + \mathbf{E}_{\mathcal{S}} J(3,t+D(1,3),\mathcal{S}))
\end{aligned} \tag{C.3}$$

To solve the full problem, we would need to do this calculation for every port, with every cargo combination at every time in the future.

C.3 Ranking algorithm example

We now show how to apply the ranking algorithm to the same example from section C.2. Suppose we are a ship currently heading to port 1, and will make a decision about which cargo to take when we arrive at time t. We do not know cargoes are

available until this time, but we can take the expectation over sets of cargoes we may observe to calculate $V(1,t) = \mathbf{E}_{\mathcal{S}} J(1, t+1, \mathcal{S})$.

1. First we calculate $Q(1,1,t), Q(1,2,t), Q(1,3,t)$ the value of taking cargoes to port 2, 3 or waiting at port 1. We have that $Q(1,2,t) = G(1,2,t) + V(2, t + D(1,2)), Q(1,3,t) = G(1,3,t) + V(3, t + D(1,3)), Q(1,1,t) = V(1, t+1)$.

2. Next we sort the cargoes in terms of long-run profitability. Suppose we have $Q(1,3,t) > Q(1,2,t) > Q(1,1,t)$ (taking cargo 3 is the most profitable in the long term, followed by cargo 2 and finally waiting), therefore, the ranked indices are as follows $s_1 = 3, s_2 = 2, s_1 = 1$.

3. We can now calculate the probability that the highest ranked cargo σ is the 1^{st},2^{nd} or 3^{rd} highest ranked cargo. The probability the highest ranked cargo is observed, is the probability the 3^{rd} cargo is available $P(\sigma = s_1) = p_3$. The probability the second highest cargo is observed is the probability the cargo 2 is available, but cargo 3 is not (otherwise we would take that) $P(\sigma = s_2) = p_2(1 - p_3)$. The probability we wait is $(1 - p_2)(1 - p_3)$.

4. Finally, we calculate the expected value of arriving at state i at time t, which is $V(i,t)$. This is the sum of the probabilities the k^{th} most valuable cargo is available multiplied by its expected value.

$$\begin{aligned} V(i,t) =& p_3 Q(1,2,t) + p_2(1-p_3)Q(1,2,t) + (1-p_2)(1-p_3)Q(1,1,t) \\ =& p_3(G(1,3,t) + V(3, t + D(1,3)))... \\ & ... + p_2(1-p_3)(G(1,2,t) + V(2, t + D(1,2)))... \\ & ... + (1-p_2)(1-p_3)V(1, t+1) \end{aligned} \tag{C.4}$$

Note this is the same as the value calculated for the DP in equation C.3, sectionC.2, but it was done without a complete enumeration of the every available cargo combination.

Bibliography

Abadi, M., P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al.
2016. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, Pp. 265–283.

Agra, A., M. Christiansen, R. Figueiredo, L. M. Hvattum, M. Poss, and C. Requejo
2013. The robust vehicle routing problem with time windows. *Computers and Operations Research*, 40(3):856 – 866. Transport Scheduling.

Ahuja, R. K., T. L. Magnanti, and J. B. Orlin
1993. Network flows: theory, algorithms, and applications.

Amemiya, T.
1978. The estimation of a simultaneous equation generalized probit model. *Econometrica: Journal of the Econometric Society*, Pp. 1193–1205.

Anderson, R., J. Huchette, C. Tjandraatmadja, and J. P. Vielma
2018. Strong convex relaxations and mixed-integer programming formulations for trained neural networks. *arXiv preprint arXiv:1811.01988*.

Andrews, L. C. and R. L. Phillips
2005. *Laser beam propagation through random media*, volume 152. SPIE press Bellingham, WA.

Athey, S., J. Tibshirani, and S. Wager
2016. Generalized random forests. *arXiv preprint arXiv:1610.01271*.

Austin, P. C. and E. A. Stuart
2017. Estimating the effect of treatment on binary outcomes using full matching on the propensity score. *Statistical methods in medical research*, 26(6):2505–2525.

Baardman, L., M. C. Cohen, K. Panchamgam, G. Perakis, and D. Segev
2015. Scheduling promotion vehicles to boost profits.

Balas, E.
1985. Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM Journal on Algebraic Discrete Methods*, 6(3):466–486.

Ban, G.-Y., N. El Karoui, and A. E. Lim
2016. Machine learning and portfolio optimization. *Management Science*.

Ban, G.-Y., J. Gallien, and A. Mersereau
2017. Dynamic procurement of new products with covariate information: The residual tree method.

Ban, G.-Y. and N. B. Keskin
2017. Personalized dynamic pricing with machine learning.

Barlow, G. L.
2000. Capacity management in the football industry. *Yield management: Strategies for the service industries*, Pp. 303–314.

Begen, M. A. and M. Queyranne
2011. Appointment scheduling with discrete random durations. *Mathematics of Operations Research*, 36(2):240–257.

Bennett, A. R. and A. L. Erera
2011. Dynamic periodic fixed appointment scheduling for home health. *IIE Transactions on Healthcare Systems Engineering*, 1(1):6–19.

Bent, R. and P. Van Hentenryck
2004a. Regrets only! online stochastic optimization under time constraints. In *AAAI*, volume 4, Pp. 501–506.

Bent, R. and P. Van Hentenryck
2007. Waiting and relocation strategies in online stochastic vehicle routing. In *IJCAI*, Pp. 1816–1821.

Bent, R. W. and P. Van Hentenryck
2004b. Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, 52(6):977–987.

Berbeglia, G., J.-F. Cordeau, and G. Laporte
2010. Dynamic pickup and delivery problems. *European Journal of Operational Research*, 202(1):8 – 15.

Bertsekas, D. P.
1995. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA.

Bertsekas, D. P. and J. N. Tsitsiklis
2002. *Introduction to probability*, volume 1. Athena Scientific Belmont, MA.

Bertsimas, D. and J. Dunn
2017. Optimal classification trees. *Machine Learning*, 106(7):1039–1082.

Bertsimas, D. and N. Kallus
2014. From predictive to prescriptive analytics. *arXiv preprint arXiv:1402.5481*.

Bertsimas, D. and N. Kallus
2016a. The power and limits of predictive approaches to observational-data-driven optimization. *arXiv preprint arXiv:1605.02347*.

Bertsimas, D. and N. Kallus
2016b. Pricing from observational data. *arXiv:1605.02347*.

Bertsimas, D. and N. Kallus
2016. The Power and Limits of Predictive Approaches to Observational-Data-Driven Optimization. *ArXiv e-prints*.

Bertsimas, D., A. OHair, S. Relyea, and J. Silberholz
2016. An analytics approach to designing combination chemotherapy regimens for cancer. *Management Science*, 62(5):1511–1531.

Besbes, O., R. Phillips, and A. Zeevi
2010. Testing the validity of a demand model: An operations perspective. *Manufacturing & Service Operations Management*, 12(1):162–183.

Bhat, N., V. Farias, and C. C. Moallemi
2012. Non-parametric approximate dynamic programming via the kernel method. In *Advances in Neural Information Processing Systems*, Pp. 386–394.

Biau, G.
2012. Analysis of a random forests model. *Journal of Machine Learning Research*, 13(Apr):1063–1095.

Biau, G., L. Devroye, and G. Lugosi
2008. Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research*, 9(Sep):2015–2033.

Biau, G. and E. Scornet
2016. A random forest guided tour. *Test*, 25(2):197–227.

Biggs, M. and R. Hariss
2018. Optimizing objective functions determined from random forests.

Blundell, R. and J. L. Powell
2003. Endogeneity in nonparametric and semiparametric regression models. *Econometric society monographs*, 36:312–357.

Blundell, R. W. and J. L. Powell
2004. Endogeneity in semiparametric binary response models. *The Review of Economic Studies*, 71(3):655–679.

Bojarski, M., D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al.
2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

Bornstein, B. H. and M. Rajki
1994. Extra-legal factors and product liability: The influence of mock jurors' demographic characteristics and intuitions about the cause of an injury. *Behavioral Sciences & the Law*, 12(2):127–147.

Bouchet, A., M. Troilo, and B. R. Walkup
2016. Dynamic pricing usage in sports for revenue management. *Managerial Finance*, 42(9):913–921.

Breiman, L.
1996. Bagging predictors. *Machine learning*, 24(2):123–140.

Breiman, L.
2001. Random forests. *Machine learning*, 45(1):5–32.

Breiman, L.
2004. Consistency for a simple model of random forests.

Breiman, L., J. Friedman, C. Stone, and R. Olshen
1984. *Classification and Regression Trees*, The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis.

Brent, R. P.
1971. An algorithm with guaranteed convergence for finding a zero of a function. *The Computer Journal*, 14(4):422–425.

Brønmo, G., M. Christiansen, K. Fagerholt, and B. Nygreen
2007. A multi-start local search heuristic for ship schedulingÃćÂĂÂŤa computational study. *Computers & Operations Research*, 34(3):900–917.

Caldwell, N., T. Srebotnjak, T. Wang, and R. Hsia
2013. âĂIJhow much will i get charged for this?âĂİ patient charges for top ten diagnoses in the emergency department. *PLoS One*, 8(2):e55491.

Carlsson, J. G. and E. Delage
2013. Robust partitioning for stochastic multivehicle routing. *Operations research*, 61(3):727–744.

Caruana, R. and A. Niculescu-Mizil
2006. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, Pp. 161–168. ACM.

Chen, T. and C. Guestrin
2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Pp. 785–794. ACM.

Chen, X., Z. Owen, C. Pixton, and D. Simchi-Levi
2015. A statistical learning approach to personalization in revenue management.

Chen, Z.-L. and H. Xu
2006. Dynamic column generation for dynamic vehicle routing with time windows. *Transportation Science*, 40(1):74–88.

Cheng, L. and M. Duran
2003. World-wide crude transportation logistics: a decision support system based on simulation and optimization. *Proceedings Foundations of Computer-Aided Process Operations*, 187.

Chernozhukov, V., D. Chetverikov, M. Demirer, E. Duflo, C. Hansen, W. Newey, and J. Robins
2018. Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal*, 21(1):C1–C68.

Chernozhukov, V., M. Goldman, V. Semenova, and M. Taddy
2017. Orthogonal machine learning for demand estimation: High dimensional causal inference in dynamic panels. *arXiv preprint arXiv:1712.09988*.

Christiansen, M., K. Fagerholt, B. Nygreen, and D. Ronen
2013. Ship routing and scheduling in the new millennium. *European Journal of Operational Research*, 228(3):467–483.

Cohen, M. C., I. Lobel, and R. Paes Leme
2016. Feature-based dynamic pricing.

Cordeau, J.-F., M. Gendreau, G. Laporte, J.-Y. Potvin, and F. Semet
2002. A guide to vehicle routing heuristics. *Journal of the Operational Research society*, 53(5):512–522.

De Farias, D. P. and B. Van Roy
2003. The linear programming approach to approximate dynamic programming. *Operations research*, 51(6):850–865.

Demirbilek, M., J. Branke, and A. Strauss
2019. Dynamically accepting and scheduling patients for home healthcare. *Health care management science*, 22(1):140–155.

Denove, C. F. and E. J. Imwinkelried
1995. Jury selection: An empirical investigation of demographic bias. *Am. J. Trial Advoc.*, 19:285.

Diehl, M. A., J. G. Maxcy, and J. Drayer
2015. Price elasticity of demand in the secondary market: Evidence from the national football league. *Journal of Sports Economics*, 16(6):557–575.

Dua, D. and C. Graff
2017. UCI machine learning repository.

Duran, S., J. L. Swann, and E. Yakıcı
2012. Dynamic switching times from season to single tickets in sports and entertainment. *Optimization Letters*, 6(6):1185–1206.

Edmonds, J.
1970. Submodular functions, matroids, and certain polyhedra. *Edited by G. Goos, J. Hartmanis, and J. van Leeuwen*, 11.

Elmachtoub, A. N. and P. Grigas
2017. Smart "Predict, then Optimize". *ArXiv e-prints*.

Engle, R. F.
1994. *Handbook of Econometrics, Volume 4*. North Holland.

Erskine, J.
2015. Stubhub announces hottest summer concerts with taylor swift at #1. `https://www.businesswire.com/news/home/20150528005481/en/StubHub-Announces-Hottest-Summer-Concerts-Taylor-Swift#.VdRNgpfEkTA`. Accessed: 2018-08-28.

Ferreira, K. J., B. H. A. Lee, and D. Simchi-Levi
2015. Analytics for an online retailer: Demand forecasting and price optimization. *Manufacturing & Service Operations Management*, 18(1):69–88.

Fikar, C. and P. Hirsch
2017. Home health care routing and scheduling: A review. *Computers & Operations Research*, 77:86–95.

Freund, Y. and R. E. Schapire
1995. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, Pp. 23–37. Springer.

Friedman, J., T. Hastie, and R. Tibshirani
2009. glmnet: Lasso and elastic-net regularized generalized linear models. *R package version*, 1(4).

Friedman, J. H.
2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, Pp. 1189–1232.

Friedman, J. H. et al.
1991. Multivariate adaptive regression splines. *The annals of statistics*, 19(1):1–67.

Gallego, G. and G. Van Ryzin
1994. Optimal dynamic pricing of inventories with stochastic demand over finite horizons. *Management science*, 40(8):999–1020.

Gendreau, M., F. Guertin, J.-Y. Potvin, and E. Taillard
1999. Parallel tabu search for real-time vehicle routing and dispatching. *Transportation science*, 33(4):381–390.

Gendreau, M., G. Laporte, and R. Séguin
1995. An exact algorithm for the vehicle routing problem with stochastic demands and customers. *Transportation science*, 29(2):143–155.

Godfrey, G. A. and W. B. Powell
2002a. An adaptive dynamic programming algorithm for dynamic fleet management, i: Single period travel times. *Transportation Science*, 36(1):21–39.

Godfrey, G. A. and W. B. Powell
2002b. An adaptive dynamic programming algorithm for dynamic fleet management, ii: Multiperiod travel times. *Transportation Science*, 36(1):40–54.

Guilkey, D. K., T. A. Mroz, and L. Taylor
1992. Estimation and testing in simultaneous equations models with discrete outcomes using cross section data. *Unpublished manuscript*.

Gurobi Optimization, L.
2019. Gurobi optimizer reference manual.

Hannaford-Agor, P. L., V. P. Hans, N. L. Mott, and G. T. Munsterman
2003. Evaluation of hung juries in bronx county, new york, los angeles county, california, maricopa county, arizona, and washington, dc, 2000–2001. *National Center for State Courts User Guide. Williamsburg, VA: National Center for State Courts*.

Harlfoxem
2016. House sales in king county, usa.

Hartje, R.
2004. A jury of your peers: How jury consulting may actually help trial lawyers resolve constitutional limitations imposed on the selection of juries. *Cal. WL Rev.*, 41:479.

Hooker, J.
2011. *Logic-based methods for optimization: combining optimization and constraint satisfaction*, volume 2. John Wiley & Sons.

Horvitz, D. G. and D. J. Thompson
1952. A generalization of sampling without replacement from a finite universe. *Journal of the American statistical Association*, 47(260):663–685.

Hwang, H.-S., S. Visoldilokpun, and J. M. Rosenberger
2008. A branch-and-price-and-cut method for ship scheduling with limited risk. *Transportation science*, 42(3):336–351.

Imbens, G. and J. Wooldridge
2007. Control function and related methods. *What?s new in Econometrics*.

Imbens, G. W. and D. B. Rubin
2015. *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction*. Cambridge University Press.

Javanmard, A. and H. Nazerzadeh
2016. Dynamic pricing in high-dimensions. *arXiv preprint arXiv:1609.07574*.

Jeroslow, R. G.
1987. Representability in mixed integer programmiing, i: characterization results. *Discrete Applied Mathematics*, 17(3):223–243.

Jeroslow, R. G. and J. K. Lowe
1984. Modelling with integer variables. In *Mathematical Programming at Oberwolfach II*, Pp. 167–184. Springer.

Jiaqi Xu, J., P. S. Fader, and S. Veeraraghavan
2019. Designing and evaluating dynamic pricing policies for major league baseball tickets. *Manufacturing & Service Operations Management*.

Ke, G., Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu
2017. Lightgbm: A highly efficient gradient boosting decision tree. In *NIPS*.

Kemper, C. and C. Breuer
2016. How efficient is dynamic pricing for sport events? designing a dynamic pricing model for bayern munich. *International Journal of Sport Finance*, 11(1).

Kleywegt, A. J., A. Shapiro, and T. Homem-de Mello
2002. The sample average approximation method for stochastic discrete optimization. *SIAM J. on Optimization*, 12(2):479–502.

Koeleman, P. M., S. Bhulai, and M. van Meersbergen
2012. Optimal patient and personnel scheduling policies for care-at-home service facilities. *European Journal of Operational Research*, 219(3):557–563.

Kohl, N., J. Desrosiers, O. B. Madsen, M. M. Solomon, and F. Soumis
1999. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1):101–116.

Kong, F. W., D. Kuhn, and B. Rustem
2010. A cutting-plane method for mixed-logical semidefinite programs with an application to multi-vehicle robust path planning. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, Pp. 1360–1365. IEEE.

Korsvik, J. E., K. Fagerholt, and G. Laporte
2010. A tabu search heuristic for ship routing and scheduling. *Journal of the Operational Research Society*, 61(4):594–603.

Kourou, K., T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis
2015. Machine learning applications in cancer prognosis and prediction. *Computational and Structural Biotechnology Journal*, 13:8 – 17.

Lagoudakis, M. G. and R. Parr
2002. Model-free least-squares policy iteration. In *Advances in neural information processing systems*, Pp. 1547–1554.

Lange, S., T. Gabel, and M. Riedmiller
2012. Batch reinforcement learning. In *Reinforcement learning*, Pp. 45–73. Springer.

Lanzarone, E. and A. Matta
2014. Robust nurse-to-patient assignment in home care services to minimize overtimes under continuity of care. *Operations Research for Health Care*, 3(2):48–58.

Lanzarone, E., A. Matta, and E. Sahin
2012. Operations management applied to home care services: the problem of assigning human resources to patients. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 42(6):1346–1363.

Lau, H. C., M. Sim, and K. M. Teo
2003. Vehicle routing problem with time windows and a limited number of vehicles. *European journal of operational research*, 148(3):559–569.

Lee, L.-F.
1981. Simultaneous equations models with discrete and censored dependent variables. *Structural analysis of discrete data with econometric applications*, 346.

Levi, R., R. O. Roundy, and D. B. Shmoys
2007. Provably near-optimal sampling-based policies for stochastic inventory control models. *Mathematics of Operations Research*, 32(4):821–839.

Ma, W., D. Simchi-Levi, and J. Zhao
2018. Dynamic pricing under a static calendar. *Available SSRN 3251015*.

Magnanti, T. L. and R. T. Wong
1981. Accelerating benders decomposition: Algorithmic enhancement and model selection criteria. *Operations research*, 29(3):464–484.

Magnanti, T. L. and R. T. Wong
1984. Network design and transportation planning: Models and algorithms. *Transportation science*, 18(1):1–55.

Malliappi, F., J. A. Bennell, and C. N. Potts
2011. A variable neighborhood search heuristic for tramp ship scheduling. In *Computational Logistics*, Pp. 273–285. Springer.

Meloni, S.
2018. Top 10 tips for tackling the growing world of big data.

Mercier, A., J.-F. Cordeau, and F. Soumis
2005. A computational study of benders decomposition for the integrated aircraft routing and crew scheduling problem. *Computers & Operations Research*, 32(6):1451 – 1476.

Mišić, V. V.
2017. Optimization of Tree Ensembles. *ArXiv e-prints*.

Mnih, V., K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller
2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al.
2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529.

Moran, G. and J. C. Comfort
1982. Scientific juror selection: Sex as a moderator of demographic and personality predictors of impaneled felony juror behavior. *Journal of Personality and Social Psychology*, 43(5):1052.

Morrice, D. J., D. Wang, J. F. Bard, L. K. Leykum, S. Noorily, and P. Veerapaneni
2014. A patient-centered surgical home to improve outpatient surgical processes of care and outcomes. *IIE Transactions on Healthcare Systems Engineering*, 4(3):119–134.

Moselle, B.
2015. National building cost manual. *Carlsbad, CA: Craftsman Book Company*.

Murthy, S. K., S. Kasif, and S. Salzberg
1994. A system for induction of oblique decision trees. *Journal of artificial intelligence research*, 2:1–32.

Nie, X. and S. Wager
2017. Learning objectives for treatment effect estimation. *preprint arXiv:1712.04912*.

Novoa, C. and R. Storer
2009. An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. *European journal of operational research*, 196(2):509–515.

Oprescu, M., V. Syrgkanis, and Z. S. Wu
2018. Orthogonal random forest for heterogeneous treatment effect estimation. *arXiv preprint arXiv:1806.03467*.

Patrick, J., M. L. Puterman, and M. Queyranne
2008. Dynamic multipriority patient scheduling for a diagnostic resource. *Operations research*, 56(6):1507–1525.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al.
2011a. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay
2011b. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pillac, V., C. Guéret, and A. Medaglia
2011. Dynamic vehicle routing problems: state of the art and prospects.

Powell, W. B.
1987. An operational planning model for the dynamic vehicle allocation problem with uncertain demands. *Transportation Research Part B: Methodological*, 21(3):217 – 232.

Powell, W. B.
2007. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons.

Powell, W. B., B. Bouzaiene-Ayari, and H. P. Simao
2007a. Dynamic models for freight transportation. *Handbooks in operations research and management science*, 14:285–365.

Powell, W. B., B. Bouzai?ne-Ayari, and H. P. SimÃčo
2007b. Dynamic models for freight transportation.

Powell, W. B. and T. A. Carvalho
1998. Dynamic control of logistics queueing networks for large-scale fleet management. *Transportation Science*, 32(2):90–109.

Powell, W. B., Y. Sheffi, K. S. Nickerson, K. Butterbaugh, and S. Atherton
1988. Maximizing profits for north american van lines' truckload division: A new framework for pricing and operations. *Interfaces*, 18(1):21–41.

Powell, W. B. and H. Topaloglu
2003. Stochastic programming in transportation and logistics. *Handbooks in operations research and management science*, 10:555–635.

Powell, W. B. and H. Topaloglu
2005. Fleet management. *Applications of Stochastic Programming*, 5:185–215.

Puterman, M. L.
2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Quinlan, J. R.
1986. Induction of decision trees. *Machine learning*, 1(1):81–106.

Ram, A., R. Prasad, C. Khatri, A. Venkatesh, R. Gabriel, Q. Liu, J. Nunn, B. Hedayatnia, M. Cheng, A. Nagar, et al.
2018. Conversational ai: The science behind the alexa prize. *arXiv preprint arXiv:1801.03604*.

Raudenbush, S. and A. S. Bryk
1986. A hierarchical model for studying school effects. *Sociology of education*, Pp. 1–17.

Reiersøl, O.
1945. *Confluence analysis by means of instrumental sets of variables*. PhD thesis, Almqvist & Wiksell.

Rivers, D. and Q. H. Vuong
1988. Limited information estimators and exogeneity tests for simultaneous probit models. *Journal of econometrics*, 39(3):347–366.

Robinson, P. M.
1988. Root-n-consistent semiparametric regression. *Econometrica: Journal of the Econometric Society*, Pp. 931–954.

Rockafellar, R. T.
2015. *Convex analysis*. Princeton university press.

Rogozhnikov, A. and T. Likhomanenko
2017. Infiniteboost: building infinite ensembles with gradient descent. *arXiv preprint arXiv:1706.01109*.

Rossum, G.
1995. Python reference manual. Technical report, Amsterdam, The Netherlands, The Netherlands.

Roy, N. and A. McCallum
2001. Toward optimal active learning through monte carlo estimation of error reduction. *ICML, Williamstown*, Pp. 441–448.

Rudin, C. and G.-Y. Vahn
2014. The big data newsvendor: Practical insights from machine learning.

Sainam, P., S. Balasubramanian, and B. L. Bayus
2010. Consumer options: Theory and an empirical application to a sports market. *Journal of Marketing Research*, 47(3):401–414.

Salzberg, S. L.
1994. C4. 5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. *Machine Learning*, 16(3):235–240.

Santibáñez, P., M. Begen, and D. Atkins
2007. Surgical block scheduling in a system of hospitals: an application to resource and wait list management in a british columbia health authority. *Health care management science*, 10(3):269–282.

Santoso, T., S. Ahmed, M. Goetschalckx, and A. Shapiro
2005. A stochastic programming approach for supply chain network design under uncertainty. *European Journal of Operational Research*, 167(1):96–115.

Scornet, E.
2014. On the asymptotics of random forests. *ArXiv e-prints*.

Scornet, E., G. Biau, J.-P. Vert, et al.
2015. Consistency of random forests. *The Annals of Statistics*, 43(4):1716–1741.

Seabold, S. and J. Perktold
2010. Statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*.

Shaked, M. and J. G. Shanthikumar
2007. *Stochastic orders*. Springer Science & Business Media.

Shotton, J., T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore
2013. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124.

Silver, D., A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al.
2016. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484.

Sönmez, T. and M. U. Ünver
2017. Market design for living-donor organ exchanges: An economic policy perspective. *Oxford Review of Economic Policy*, 33(4):676–704.

Stock, J. H. and M. W. Watson
2015. *Introduction to econometrics*.

Strobl, C., A.-L. Boulesteix, T. Kneib, T. Augustin, and A. Zeileis
2008. Conditional variable importance for random forests. *BMC bioinformatics*, 9(1):307.

Sweeting, A.
2012. Dynamic pricing behavior in perishable goods markets: Evidence from secondary markets for major league baseball tickets. *Journal of Political Economy*, 120(6):1133–1172.

Talluri, K. T. and G. J. Van Ryzin
2006. *The theory and practice of revenue management*, volume 68. Springer Science & Business Media.

Tang, L., W. Jiang, and G. K. Saharidis
2013. An improved benders decomposition algorithm for the logistics facility location problem with capacity expansions. *Annals of operations research*, 210(1):165–190.

Tirado, G., L. M. Hvattum, K. Fagerholt, and J.-F. Cordeau
2013. Heuristics for dynamic and stochastic routing in industrial shipping. *Computers & Operations Research*, 40(1):253–263.

Topaloglu, H. and W. B. Powell
2003. An algorithm for approximating piecewise linear concave functions from sample gradients. *Operations Research Letters*, 31(1):66–76.

Topaloglu, H. and W. B. Powell
2006. Dynamic-programming approximations for stochastic time-staged integer multicommodity-flow problems. *INFORMS Journal on Computing*, 18(1):31–42.

Torres-Ramos, A., E. Alfonso-Lizarazo, L. Reyes-Rubiano, and C. Quintero-Araújo
2014. Mathematical model for the home health care routing and scheduling problem with multiple treatments and time windows. In *Proceedings of the 1st International Conference on Mathematical Methods & Computational Techniques in Science & Engineering*, number s 140, P. 145.

Varian, H. R.
2014. Big data: New tricks for econometrics. *The Journal of Economic Perspectives*, 28(2):3–27.

Vielma, J. P.
2018. Small and strong formulations for unions of convex sets from the cayley embedding. *Mathematical Programming*, Pp. 1–33.

Wager, S.
2014. Asymptotic theory for random forests. *arXiv preprint arXiv:1405.0352*.

Wager, S. and S. Athey
2017. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*.

Wang, X., V.-A. Truong, and D. Bank
2018. Online advance admission scheduling for services with customer preferences. *arXiv preprint arXiv:1805.10412*.

Wright, P. G.
1928. *Tariff on animal and vegetable oils*. Macmillan Company, New York.

Xie, Y., J. E. Brand, and B. Jann
2012. Estimating heterogeneous treatment effects with observational data. *Sociological methodology*, 42(1):314–347.

Yalçındağ, S., A. Matta, and E. Şahin
2012. Operator assignment and routing problems in home health care services. In *2012 IEEE International Conference on Automation Science and Engineering (CASE)*, Pp. 329–334. IEEE.

Yuan, B., R. Liu, and Z. Jiang
2015. A branch-and-price algorithm for the home health care scheduling and routing problem with stochastic service times and skill requirements. *International Journal of Production Research*, 53(24):7450–7464.

Zhu, J.-D.
2014. Effect of resale on optimal ticket pricing: Evidence from major league baseball tickets. Technical report, Working paper, Texas A&M University.

Zou, H. and T. Hastie
2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.