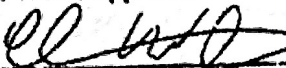


PhD Thesis Approval

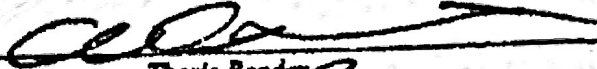
THESIS TITLE: *Measuring the Algorithms in Online Marketplaces*

AUTHOR: *Le CHEN*

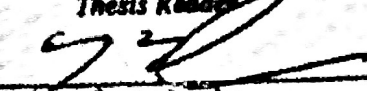
Ph.D. Thesis Approved to complete all degree requirements for the Ph.D. Degree in Computer Science.


Thesis Advisor


5/6/17
Date


Thesis Reader

4/24/17
Date


Thesis Reader

4/24/17
Date



Thesis Reader

04/24/2017
Date

Thesis Reader


Date

GRADUATE SCHOOL APPROVAL:


Director, Graduate School

5/15/2012
Date

COPY RECEIVED IN GRADUATE SCHOOL OFFICE:


Recipient's Signature

5/15/2017
Date

Distribution: Once completed, this form should be scanned and attached to the front of the electronic dissertation document (page 1). An electronic version of the document can then be uploaded to the Northeastern University-UMI website.

Measuring Algorithms in Online Marketplaces

A Dissertation Presented

by

Le Chen

to

The College of Computer and Information Science

in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

in

Computer Science

**Northeastern University
Boston, Massachusetts**

May 2017

ProQuest Number: 10283491

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



ProQuest 10283491

Published by ProQuest LLC (2017). Copyright of the Dissertation is held by the Author.

All rights reserved.

This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Dedicated to my courageous and lovely wife.

Contents

List of Figures	v
List of Tables	viii
Acknowledgments	x
Abstract of the Dissertation	xi
1 Introduction	1
1.1 Uber	2
1.2 Amazon Marketplace	4
1.3 Hiring Sites	7
1.4 Outline	9
2 Related Work	10
2.1 Online Marketplaces	10
2.1.1 Price Competition	10
2.1.2 Fraud and Privacy	11
2.2 Hiring Discrimination	11
2.2.1 The Importance of Rank	12
2.3 Auditing Algorithms	13
2.3.1 Filter Bubbles	13
2.3.2 Price Discrimination	14
2.3.3 Gender and Racial Discrimination	14
2.3.4 Privacy and Transparency	15
3 Uber’s Surge Pricing Algorithm	16
3.1 Background	16
3.2 Methodology	18
3.2.1 Selecting Locations	19
3.2.2 The Uber API	19
3.2.3 Collecting Data from the Uber App	19
3.2.4 Calibration	22
3.2.5 Validation	24

3.3	Analysis	25
3.3.1	Data Collection and Cleaning	26
3.3.2	Dynamics Over Time	27
3.3.3	Spatial Dynamics and EWT	29
3.4	Surge Pricing	30
3.4.1	The Cost of Surges	31
3.4.2	Surge Duration and Updates	32
3.4.3	Surge Areas	33
3.4.4	Algorithm Features and Forecasting	34
3.4.5	Impact of Surge on Supply and Demand	36
3.5	Avoiding Surge Pricing	38
3.6	Summary	40
4	Algorithmic Pricing on Amazon Marketplace	42
4.1	Background	42
4.1.1	Amazon Marketplace	43
4.1.2	The Buy Box	45
4.1.3	Amazon Marketplace Web Service	46
4.2	Data Collection	46
4.2.1	Obtaining Sellers and Prices	46
4.2.2	Crawling Methodology	47
4.2.3	Selecting Products	49
4.3	The Buy Box	51
4.3.1	Seller Ranking	51
4.3.2	Behavior of the Buy Box Algorithm	51
4.3.3	Algorithm Features and Weights	52
4.4	Dynamic Pricing Detection	56
4.4.1	Methodology	56
4.4.2	Algorithmic Pricing Sellers	58
4.4.3	Price Matching Examples	60
4.5	Analysis	60
4.5.1	Business Practices	61
4.5.2	Competition	64
4.5.3	Amazon as a Seller	66
4.6	Simulations	67
4.6.1	Methodology	68
4.6.2	Simulation Results	70
4.6.3	Breaking Into the Buy Box	74
4.7	Summary	76
5	Impact of Gender on Hiring Sites	77
5.1	Background	77
5.1.1	Resume Search Engines	79
5.1.2	Hiring Websites	80
5.2	Data Collection	83

5.3	Analysis	86
5.3.1	Ranking Bias	86
5.3.2	Fairness	91
5.3.3	Uncovering Hidden Features	101
5.4	Alternative Ranking Approaches	102
5.4.1	Algorithms	103
5.4.2	Evaluation	104
5.5	Summary	110
6	Conclusion	112
6.1	Summary and Impact	112
6.2	Ethics for Algorithm Audits	114
6.3	A Guide for Future Audits	116
6.3.1	Incentive Misalignment	116
6.3.2	Biases in Human Data	117
	Bibliography	119

List of Figures

3.1	Screenshot of the Uber Partner app.	18
3.2	Visibility radius of clients in two cities.	23
3.3	Locations of my Uber and taxi measurement points in SF and Manhattan.	24
3.4	Measured and ground-truth supply (left) and demand (right) of taxis in midtown Manhattan.	25
3.5	Lifespan of UberX cars in Manhattan and SF before data cleaning.	27
3.6	Distance of short-lived insiders to the boundary.	27
3.7	Lifespan of different types of Uber cars after data cleaning.	27
3.8	Supply, demand, surge multiplier, and EWT over time for midtown Manhattan and downtown SF. Surge multiplier and estimated wait time (EWT) are only shown for UberX. Diurnal patterns are observed in supply and demand, but the characteristics of the surge multiplier show less predictability.	28
3.9	Heatmaps for UberX cars in Manhattan.	30
3.10	Heatmaps for UberX cars in SF.	30
3.11	Distribution of EWTs for UberXs.	31
3.12	Distribution of surge multipliers for UberXs.	31
3.13	Duration of surges for UberXs.	31
3.14	Examples of surge over time as seen from (a) the API and (b) the Client app. Although surge is recalculated every 5 minutes in both cases, clients also observe surge jitters for 20-30 seconds (red arrows).	31
3.15	Moment when surge changes during each interval.	33
3.16	Distribution of surge multipliers seen during times of jitter.	33
3.17	Fraction of clients that observe jitter at the same time.	33
3.18	Surge areas in in Manhattan.	35
3.19	Surge areas in SF.	35
3.20	(Supply - Demand) vs. Surge for UberX.	35
3.21	EWT vs. Surge for UberX.	35
3.22	Transition probabilities of UberXs when all areas have equal surge, and when one area is surging.	37
3.23	Fraction of the time when passengers would receive lower surge prices on UberXs by walking to an adjacent area.	39
3.24	Amount that surge is reduced and walking time needed when passengers walk to an adjacent area.	39

4.1	An example Buy Box on Amazon.	43
4.2	An example <i>New Offers</i> page on Amazon, listing all sellers for a given product. . .	45
4.3	Frequency of page updates.	48
4.4	Examples of price jitter (highlighted with arrows) in the Buy Box on a product page.	48
4.5	Cumulative distribution of product prices.	50
4.6	Cumulative distribution of number of sellers per product.	50
4.7	Correlation between price and rank (-1 is anti-correlation, 1 is perfect correlation).	50
4.8	Cumulative distribution of changes to the Buy Box price and winner, per product. .	52
4.9	Probability of winning the Buy Box for sellers at different ranks.	52
4.10	Buy Box winner prediction accuracy for products with different numbers of sellers.	52
4.11	Number of algorithmic sellers detected with different change thresholds.	57
4.12	CDF of absolute price differences between algorithmic sellers and target prices. . .	57
4.13	CDF of relative price differences between algorithmic sellers and target prices. . .	57
4.14	Example of 3P seller (in red) matching the lowest price of all other sellers.	59
4.15	A second example of 3P seller (in red) matching the lowest prices offer by two other sellers.	59
4.16	Example of Amazon (in red) setting a premium over the lowest price of all other sellers.	59
4.17	Example of Amazon (in red) matching to the lowest price over time.	59
4.18	Distribution of seller/product lifetimes for different types of sellers.	62
4.19	Number of unique products offered for sale by different types of sellers.	62
4.20	Percentage of positive customer feedback received by different types of sellers. . .	62
4.21	Amount of feedback received by different types of sellers.	63
4.22	Cumulative distribution of rank on the <i>New Offers</i> page for different types of sellers.	63
4.23	Number of sellers of the same type per product.	63
4.24	Number of price changes per seller/product pair.	65
4.25	Number of changes in the Buy Box seller for products with and without algorithmic and Prime sellers.	65
4.26	Number of changes in the Buy Box price for products with and without algorithmic and Prime sellers.	65
4.27	Probability of winning the Buy Box for different types of sellers at different ranks.	67
4.28	Cumulative distribution of Amazon's rank in the presence/absence of algorithmic and Prime sellers.	67
4.29	Percentage of checkouts amassed by each seller under four different customer behav- ior models.	70
4.30	Expected revenue earned by each seller under four different customer behavior models.	70
4.31	Percentage of checkouts per seller for algorithmic and non-algorithmic sellers, under four different models of customer behavior.	72
4.32	Revenue per seller for algorithmic and non-algorithmic sellers, under four different models of customer behavior.	73
4.33	Distribution of sellers across the hyperplane in my logistic regression classifier. . .	75
4.34	Price change required for sellers to win the Buy Box, relative to three baselines. . .	75
5.1	Screenshots of search results on Indeed and Monster when searching for a "Software Engineer" near New York City.	78
5.2	Inferred probability of being male for all candidates in my dataset.	85

5.3	Recall of being female for candidates when searching for a “Software Engineer”. The 20 lines correspond to different cities. Lines below the diagonal mean that fewer female candidates appear than expected, given the overall population of candidates. Lines above the diagonal indicate the opposite, i.e., there are fewer male candidates than expected.	87
5.4	Recall curves for “Software Engineer” in three cities on Monster. All three curves show a discernable difference between the rank of female and male candidates, but this is not always detected by the Mann-Whitney U test.	88
5.5	nDCG comparison of the predicted rankings $\hat{\mathbf{R}}$ produced by the mixed linear models versus the original rankings \mathbf{R} . These results were computed using the Original and Matched candidates from my dataset.	92
5.6	nDCG comparison of the unbiased and neutral rankers when compared to the original search results. Figures 5.6a and 5.6b focus on the top 1000 candidates, while Figures 5.6c and 5.6d focus on the top 100.	105
5.7	Change in the median rank of female candidates after applying my unbiased and neutral rankers. Positive values correspond to increases in the median rank for female candidates after re-ranking. Figures 5.7a and 5.7b focus on the top 1000 candidates, while Figures 5.7c and 5.7d focus on the top 100.	106
5.8	Change in the gender gap after applying my unbiased and neutral rankers. Positive values correspond to shrinking gender gaps, i.e., the median rank between male and female candidates is getting smaller after re-ranking. Figures 5.8a and 5.8b focus on the top 1000 candidates, while Figures 5.8c and 5.8d focus on the top 100.	107
5.9	Change in the value of the <i>Experience</i> and <i>Education</i> feature after applying my unbiased and neutral rankers on the top 100 Software Engineer candidates in New York City. Red dots are for the unbiased ranker, while blue dots are for the neutral ranker, as shown in Figure 5.9b.	109

List of Tables

3.1	R^2 scores of linear regression in Manhattan and SF	35
4.1	Relative importance of different features in winning the Buy Box, as determined by my DT classifier. The significance of coefficients in the regression model are: $*p < 0.05$; $**p < 0.01$; $***p < 0.001$	54
4.2	Number of sellers and products with detected algorithmic pricing, based on two different change thresholds. I use a change threshold of 20 unless otherwise stated.	58
4.3	Percentage of products with Amazon as one of the listed sellers.	67
5.1	Per-candidate features I extract from search results on Indeed , Monster , and CareerBuilder . I infer the last feature, <i>Gender</i> , based on each candidate’s first name; other features are directly observed. Note that not all features are present on all hiring websites.	82
5.2	Job titles used in my queries, organized by EEOC category.	83
5.3	Cities used in my queries.	83
5.4	Mann-Whitney U test on the rank of females and males in the top 1000 candidates per job title. Each cell shows the percentage (%) of cities with $p < 0.05$ for a given job title on a given hiring website. Cases where the percentage is $\geq 20\%$ are bolded . I show results for the original search results, as well as search results that have been re-ranked using my unbiased and neutral ranking algorithms. Note that I skip samples when the number of female or male candidates is less than 20; “–” marks instances where there are no cities with sufficiently large populations to test. I also remove the 8% of candidates with ambiguous genders.	90
5.5	Estimated coefficients and standard deviation of mixed linear regressions on the top 1000 and top 100 Original and Matched candidates in search results from each hiring website, grouped by city and job title. Significance level is unavailable for <i>Random Effect</i> . Note: $*p < 0.05$; $**p < 0.01$; $***p < 0.001$	93
5.6	Estimated <i>Probability of Being Male</i> coefficient from mixed linear regressions as the length of the search result list is varied. These results were computed using the Original candidates from my dataset. Note: $*p < 0.05$; $**p < 0.01$; $***p < 0.001$	94
5.7	Coarsened Exact Matching statistics for all three hiring websites on the top 1000 candidates. Note that <i>Rank</i> is not a feature in the matching procedure; I list it here to check the balance statistics. The “Binning” column shows the type of each feature (categorical or numerical), how many bins were used, and whether those bins were chosen Automatically by MatchIt or Manually by me.	98

5.8	Coarsened Exact Matching statistics for all three hiring websites on the top 100 candidates. Note that <i>Rank</i> is not a feature in the matching procedure; I list it here to check the balance statistics. The “Binning” column shows the type of each feature (categorical or numerical), how many bins were used, and whether those bins were chosen A utomatically by MatchIt or M anually by me.	99
5.9	Means, standard deviation, and correlations of the variables on Indeed, Monster, and CareerBuilder. These results were computed using the O riginal candidates from my dataset.	100
5.10	Features tested in my controlled resume experiments. Cases where candidate <i>A</i> ranks higher than <i>B</i> reveals that the given feature is taken into account by the ranking algorithm.	102

Acknowledgments

First and foremost, I want to express my heartfelt gratitude to my advisor, Christo Wilson. I am extremely fortunate to have an advisor who gives me tremendous freedom to explore different disciplines, who teaches me how to conduct fine research with all his knowledge, patience and passion, and on top of all, who advises me in many aspects of my life path. None of my work would have been remotely possible without his countless help and support. His wisdom, kindness, and courage guided me all the way to where I am now, and will continue to shape my path in the future.

I am extremely grateful to Alan for giving me invaluable ideas and advice on the exciting projects, for the optimism and enthusiasm he shows in every meeting and discussion, and for his mid-night emergency meetings fixing bugs before the paper deadlines.

I am very thankful to Ben for his useful and encouraging advice about PhD experience, research goals, and career path. Some of the advice is from meeting him in person, while the rest comes from his informative Quora answers.

I also sincerely thank my committee member Chris, for vetting my interdisciplinary study, and providing useful suggestions and ideas to make the work stronger and more rigorous.

I am very grateful to all the awesome labmates and collaborators: Ahmad, Shan, Ancsa, Liang, Yabing, Arash, Fangfan, Jingjing, Ruijun, and Priyam, whose support help me overcome many obstacles in all these years. Also I want to thank Dave and Cristina for all the insightful suggestions to my thesis, which make my work more clear and generalized.

Finally, and most importantly, I give all my thanks to my family. Thanks to my parents for always being there for me. Thanks to my wife, Xizhe: I could not have been here without your unconditional love and support during the journey.

Abstract of the Dissertation

Measuring Algorithms in Online Marketplaces

by

Le Chen

Doctor of Philosophy in Computer Science

Northeastern University, May 2017

Dr. Christo Wilson, Adviser

The integration of Big Data and algorithms have revolutionized many aspects of modern life. One of the revolutions is happening in markets, where data-centric algorithms are gradually automating services that used to require manual operations. For example, Real-Time Bidding (RTB) algorithms are widely adopted in the online advertising network, and are expected to grow from generating 31% of the revenue of the entire digital ads market in 2015 to 48% in 2020. Mortgage and credit lenders in the loan markets use algorithms to make lending decisions based on Big Data rather than traditional credit reports. Outside of advertising and financial markets, ride sharing companies like Uber and Lyft use surge pricing algorithms to dynamically balance the gaps between car supply and user demand. Sellers in e-commerce markets use dynamic pricing tools to adjust product prices and manage inventories in real-time. Surveys show that as of 2013, 13% of retailers had deployed dynamic pricing algorithms.

Algorithms are powerful tools that have the potential to improve the efficiency of markets, but come at a cost of possible harms. On one hand, algorithms can be beneficial: RTB maps billions of Internet users to customized advertisements based on their interests in real time, and thus increases the effectiveness for both advertisers and publishers in terms of advertising inventory sold. Similarly, ride sharing has redefined the transportation market by matching millions of drivers and riders around the globe in real-time. On the other hand, evidence shows that problems may be caused by algorithms, such as racial discrimination on Google's advertisement services, or unpredictable prices shown to users in online marketplaces.

Unfortunately, we currently lack measurement tools or methodologies to *audit* the behavior of these algorithms. As a result, we are unable to measure the impact of these algorithms on people. For example, the general public is unable to access the details about how credit scores and prices are calculated in online marketplaces, since algorithms are typically proprietary trade secrets. Similarly,

for Machine Learning algorithms, the data for training the predicative models may also be unavailable. The lack of transparency surrounding algorithms and the data that powers them has led to concerns about whether they are being manipulated by companies to increase their own profit, and whether they are fair and unbiased to users.

The goal of my work is to develop methodologies and build measurement tools to audit and understand the impact of algorithms in online marketplaces. I focus on three types of marketplaces: the ride sharing marketplace Uber, the e-commerce marketplace Amazon, and human labor marketplaces Indeed, Monster, and CareerBuilder. Algorithms play crucial roles on all these platforms, and potential fairness and manipulation issues caused by the algorithms may be present in these systems.

First, I examine Uber’s surge pricing algorithm to answer questions such as whether ride prices are true reflections of supply and demand dynamics, and whether surge prices can be manipulated by the company or passengers. I gather four weeks of data from Uber by emulating 43 copies of the Uber smartphone app and distributing them throughout downtown San Francisco (SF) and midtown Manhattan. Using my dataset, I am able to characterize the dynamics of Uber in SF and Manhattan, as well as identify key implementation details of Uber’s surge price algorithm. My observations about Uber’s surge price algorithm raise important questions about the fairness and transparency of this system.

Next, I investigate two major and correlated algorithmic components on Amazon Marketplace that determine the product prices paid by consumers: the Buy Box and dynamic pricing by sellers in the market. In this study, I first conduct an in-depth investigation on the features and weights that drive the Buy Box algorithm. Then I develop a methodology for detecting dynamic pricing by sellers, and use it to empirically analyze their prevalence and behavior on Amazon Marketplace. I gather four months of data covering all merchants selling any of 1,641 best-seller products. Using this dataset, I am able to uncover the algorithmic pricing strategies adopted by over 500 sellers. I then explore the characteristics of these sellers and characterize the impact of these strategies on the dynamics of the marketplace.

Finally, I study the ranking algorithms that power resume search engines on hiring websites, and investigate gender-based inequalities in their search results. I collect search results from Indeed, Monster, and CareerBuilder based on 35 job title queries in 20 American cities, resulting in data on over 855K job candidates. Using statistical tests and regression analysis, I find statistically significant evidence of two types of inequality on all three websites (ranking bias and unfairness), almost always to the detriment of female candidates. Motivated by these findings, I propose two alternative ranking methods that encode different definitions of fairness, and examine the inherent tradeoffs posed by trying to achieve gender-fairness in hiring markets.

Altogether, my work presents techniques to measure the impact of algorithms in online marketplaces. My methods can be extended to other platforms and services, in order to increase transparency and provide insights into how these systems affect people. Ultimately, I hope that my research helps people to find and mitigate issues present in opaque algorithmic systems.

Chapter 1

Introduction

The integration of Big Data and algorithms have revolutionized many aspects of modern life. One of the revolutions is happening in markets, where data-centric algorithms are gradually automating services that used to require manual operations. For example, Real-Time Bidding (RTB) algorithms [54, 191, 192] are widely adopted in online advertising networks, and are expected to grow from generating 31% of the revenue of the entire digital ads market in 2015 to 48% in 2020 [14]. In the loan market, mortgage and credit lenders use algorithms to make lending decisions based on Big Data rather than traditional credit reports [123]. Outside of advertising and financial markets, ride sharing companies like Uber and Lyft use surge pricing algorithms to dynamically balance car supply and user demand [2, 3]. In e-commerce markets, sellers use dynamic pricing tools to adjust product prices and manage inventories in real-time [93]. Surveys show that as of 2013, 13% of retailers had deployed dynamic pricing algorithms [26].

On the one hand, algorithms are powerful tools that have the potential to improve the efficiency of markets. For example, RTB maps billions of Internet users to customized advertisements based on their interests in real-time, and thus increases the effectiveness for both advertisers and publishers in terms of advertising inventory sold [167]. Similarly, ride sharing companies have redefined the transportation market by expanding beyond the fixed limit of taxi supply in each city and matching millions of drivers and passengers around the globe in real-time.

On the other hand, algorithms may introduce harms due to design flaws. For example, products with unexpected prices may be shown to users in online marketplaces: two competing dynamic pricing algorithms inadvertently raised the price of a used textbook to \$23M on Amazon [175]. In ride sharing services, concerns about the Uber surge pricing algorithms [80] were exacerbated when Uber was forced to publicly apologize and refund rides after prices surged during Hurricane

CHAPTER 1. INTRODUCTION

Sandy [106] and the Sydney hostage crisis [128]. As of April 2016, Uber was getting sued for allegedly using surge pricing to illegally fix prices [111].

Furthermore, for Machine Learning based algorithms, training data generated by real people may encode human-biases that may also cause harms. For example, racial discrimination was found on Google’s advertisement services: Sweeney showed that there is a positive correlation between searching for African American names and returning arrest record advertisements [176]. Similarly, Amit et al. designed an automated tool that explored the interaction of user behaviors, Google’s ads, and Ad Preferences, and conducted experiments to uncover gender discrimination on Google’s ad service [64].

The potential harms of algorithms and Big Data have drawn attention from governmental regulators. The White House released a report warning that certain Big Data analytic techniques being used by businesses may discriminate against customers [126]. The Federal Trade Commission (FTC) held a public workshop to explore the potential impact of Big Data on low-income and under-served populations [15]. The European Parliament conducted a study to investigate price discrimination by e-commerce sites against consumers based on place of residence or nationality [4].

Although both the general public and regulators are aware of the possible harms algorithms may bring to people, unfortunately there is a lack of measurement tools and methodologies to *audit* the behavior of these algorithms. As a result, we are unable to measure the impact of these algorithms on people. For example, the general public and regulators are unable to access the details about how credit scores and prices are calculated in online marketplaces, since algorithms are typically proprietary trade secrets. Similarly, for Machine Learning algorithms, the data for training the predicative models may also be unavailable.

In this thesis, I aim to address this situation by developing methodologies and building measurement tools to audit and understand the impact of algorithms in online marketplaces. My work focuses on three types of marketplaces: the ride sharing marketplace Uber [51], the e-commerce marketplace Amazon [52], and human labor marketplaces Indeed, Monster, and CareerBuilder. In particular, my thesis will make contributions outlined in the following paragraphs.

1.1 Uber

Uber has emerged as a leader in the sharing economy. Uber is a “ride sharing” service that matches willing drivers (i.e., anyone with a car) with customers looking for rides. Uber charges passengers based on time and distance of travel; however, during times of high demand, Uber uses a

CHAPTER 1. INTRODUCTION

“surge multiplier” to increase prices. Uber provides two justifications for surge pricing [84]: first, it reduces demand by pricing some customers out of the market, thus reducing the wait times for the remaining customers. Second, surge pricing increases profits for drivers, thus incentivizing more people to drive during times of high demand. Uber’s business model has become so widely emulated that the media now refers to the “Uberification” of services [35].

While Uber has become extremely popular, there are also concerns about the fairness, efficacy, and disparate impact of surge pricing. The key difference between Uber and other sharing economy marketplaces is that Uber is a black-box: they do not provide data about supply and demand, and surge multipliers are set by an opaque algorithm. This lack of transparency has led to concerns that Uber may artificially manipulate surge prices to increase profits [152], as well as apprehension about the fairness of surge pricing [81].

I present the first in-depth investigation of Uber. To understand the impact of surge pricing on passengers and drivers, I gather four weeks of data from Uber by emulating 43 copies of the Uber smartphone app and distributing them in a grid throughout downtown San Francisco and midtown Manhattan. By carefully calibrating the GPS coordinates reported by each emulated app, I am able to collect high-fidelity data about surge multipliers, estimated wait times (EWTs), car supply, and passenger demand for all types of Ubers (e.g., UberX, UberBLACK, *etc.*). I validate my methodology using ground-truth data on New York City (NYC) taxicabs.

This measured data enables me to identify key implementation details of Uber’s surge price algorithm, which also illuminates design flaws in the algorithm related to *opacity* and *incentive failure*.

Opacity. The implementation details of Uber’s surge pricing algorithm are hidden from both passengers and drivers. Unfortunately, this opacity hides flaws. For example, as discussed in § 3, I observe that Uber has manually divided cities into “surge areas” with independent surge prices. The divided regions create opportunities for exploitation by passengers, as I demonstrate how passengers can significantly reduce surge prices 10-20% of the time by walking to a neighboring surge areas. Furthermore, I discover that prices update every 5 minutes, which makes it difficult for drivers to optimize their behavior in order to capture surge revenues.

Similarly, opacity hides bugs. For example, as I show in § 3, in April 2015, Uber began serving surge prices to users that did not always match the prices returned by the Uber official API. Furthermore, surge prices were no longer uniform for users, even if they were in the same surge area at the same time. I reported this finding to Uber, and their engineers confirmed that this behavior was

CHAPTER 1. INTRODUCTION

caused by a bug in their system. This bug raised serious fairness issues in the Uber system, and was unnoticed for months.

Incentive Failure. The motivation behind surge pricing is to use high fares to incentivize more drivers to pick up passengers in the high-demand areas, as well as to temporarily reduce the passenger demand. However, by analyzing the movements and actions of Uber drivers, I show that surge prices only have a small, positive effect on vehicle supply, but a large, negative impact on passenger demand, resulting in less booked cars in the surging areas. Moreover, I observe that drivers flee from the surging areas, probably due to the decrease in demand. This contradicts the motivation of Uber’s surge pricing algorithm. An open market would not suffer from this type of failure because information about supply and demand would allow passengers and drivers to naturally reach equilibrium.

1.2 Amazon Marketplace

The rise of e-commerce has unlocked practical applications for *algorithmic pricing* (sometimes referred to as dynamic pricing algorithms or Revenue/Yield Management). Algorithmic pricing strategies are challenging to implement in traditional retail settings due to lack of data (e.g., competitors’ prices) and physical constraints (e.g., manually relabeling prices on products). In contrast, e-commerce is unconstrained by physical limitations, and collecting real-time data on customers and competitors is straightforward. Travel websites are known to use personalized pricing [89], while some e-retailers are known to automatically match competitors prices [44, 194].

While algorithmic pricing can make merchants more competitive and potentially increase revenue, it also creates new challenges. *First*, poorly implemented pricing algorithms can interact in unexpected ways and even produce unexpected results, especially in complex environments populated by other algorithms. For example, two competing dynamic pricing algorithms inadvertently raised the price of a used textbook to \$23M on Amazon [175]; reporters have noted that similar algorithmic pricing also exists in day-to-day commodities [18]. *Second*, dynamic pricing algorithms can implement collusive strategies that harm consumers. For example, the US Justice Department successfully prosecuted several individuals who implemented a price fixing scheme on Amazon using algorithms [9]. Unfortunately, regulators and the public currently lack comprehensive knowledge about the prevalence and behavior of algorithmic pricing algorithms in-the-wild.

In this study, I empirically analyze deployed algorithmic pricing strategies on Amazon

CHAPTER 1. INTRODUCTION

Marketplace. Specifically, my goal is to understand what algorithmic pricing strategies are used by participants in the market, how prevalent these strategies are, and ultimately how they impact customer experience. I chose to focus on Amazon for three reasons: *first*, Amazon is the largest e-commerce destination in the US and Europe [39]. *Second*, Amazon is an open marketplace populated by third-party sellers, as well as Amazon itself. *Third*, Amazon’s platform provides APIs that are specifically designed to facilitate algorithmic pricing [5]. *Fourth*, Amazon’s loyalty program (Amazon Prime) has been very successful: more than half of all Amazon customers are estimated to be Prime subscribers [166]. The existence of the Prime program gives me additional opportunities to investigate competition and incentives in the Amazon marketplace.

To perform this study, I develop a novel methodology to uncover sellers that are likely using algorithmic pricing. I collect five months of data from 1,641 of the most popular products on Amazon. I gather information about the top-20 sellers of each product every 25 minutes, including the sellers’ prices, ratings, whether their products are eligible for two-day Prime shipping, and other attributes. I use this data to analyze changes in price over time, as well as compare the attributes of sellers. I focus on top selling products because they tend to have multiple sellers, and thus are likely to exhibit more competitive dynamics.

I begin by analyzing the algorithm underlying Amazon’s *Buy Box*. This algorithm determines, for a given product being sold by many sellers, which of the sellers will be featured in the Buy Box on the product’s landing page (i.e., which seller is the “default” seller). As shown in Figure 4.1, customers use the Buy Box to add products to their cart; sellers not selected for the Buy Box are relegated to a separate webpage. The precise features and weights used by the Buy Box algorithm are unknown [31], yet the algorithm is of critical importance since 82% of sales on Amazon go through the Buy Box [177]. For my purposes, understanding the Buy Box algorithm is important because sellers may choose dynamic pricing strategies that maximize their chance of being selected by the algorithm.

Next, I examine the dynamic pricing strategies used by sellers in Amazon Marketplace. To identify pricing algorithms, I treat the *target price* of each product (e.g., the lowest advertised price or Amazon’s price) as a time series, and use correlative analysis to identify specific sellers whose prices track the target price over time. Overall, I identify over 500 sellers who are very likely using algorithmic pricing.

I then compare the characteristics of algorithmic and non-algorithmic sellers, as well as Prime and non-Prime sellers (i.e., those sellers who do and not offer two-day Prime shipping). I observe that 43% of algorithmic sellers offer Prime shipping, demonstrating a strong correlation

CHAPTER 1. INTRODUCTION

between the use of sophisticated inventory management and fulfillment systems.

Finally, I use simulations of different customer buying strategies to evaluate the success of sellers in the Amazon Marketplace in terms of *purchases* (i.e., how many items they will sell) and *revenue*. I observe that algorithmic sellers have a significant advantage over non-algorithmic sellers in terms of purchases and revenue because they occupy the Buy Box more frequently, and because they are often able to charge higher prices than competing sellers.

Comparing the characteristics between algorithmic and non-algorithmic sellers, I discover two fairness issues on Amazon Marketplace: *high dynamics* and *feedback loops*.

High Dynamics. Amazon allows sellers to update product price at any time. This feature adds fluidity to the market, but also introduces high dynamics. As I show in § 4, the advertised price in the Buy Box for products with algorithmic sellers is significantly more volatile than for products without any algorithmic sellers. These rapidly fluctuating prices may lead to customer dissatisfaction [18].

Feedback Loops. I observe that algorithmic (and Prime) sellers appear to be more successful than non-algorithmic (non-Prime) sellers: they offer fewer products, but receive significantly higher amounts of customer feedback, which suggests they have much higher sales volume. As I show in § 4, number of feedback is a strong feature for winning the Buy Box. This causes algorithmic sellers to “win” the Buy Box more frequently, even when they do not offer the lowest price for a given product, which may further contribute to their feedback scores. This “strong-gets-stronger” feedback loop makes the competition more difficult for sellers who do not adopt algorithmic pricing.

I also run simulations to investigate how much sellers who are not winning the Buy Box would need to lower their prices to “change their fate” and win the Buy Box. Surprisingly, I find that 85% of sellers **cannot** win the Buy Box, even if they drastically lower their prices, due to the impact of other attributes that they cannot directly control, like their customer feedback scores. In the small fraction of cases where a seller could potentially win the Buy Box by lowering their price, the average price drop necessary to win is 34%, which is likely to erase the seller’s profit margin, or even cause them to sell at a loss. Although these results are derived from simulations, they suggest that it is extremely challenging for new sellers to make a profit offering best-selling products on Amazon Marketplace.

1.3 Hiring Sites

The internet is fundamentally changing the labor economy. Millions of people use services such as LinkedIn, Indeed, Monster, and CareerBuilder to find employment [45, 101, 133]. These online services offer innovative mechanisms for recruiting and organizing employment, often driven by algorithmic systems that rate, sort, recommend, and match workers and employers.

In theory, many of the mechanisms that cause discrimination in traditional labor markets (e.g., cognitive biases and network homophily) should be absent in online services. In online contexts, workers' demographics may be less clear or even anonymized, while employers and employees find one another through demographic-agnostic algorithmic searches. For example, on Indeed, Monster, and CareerBuilder, job seekers create detailed personal profiles and upload resumes, but the services specifically do not collect demographic information or profile images. This design explicitly discourages recruiters from basing hiring decisions on demographic factors.

Yet, evidence indicates that inequalities persist in many different online labor contexts. Scholars have uncovered cases of unequal opportunities presented to women in online ads [63]; biases in social feedback for gig-economy workers based on gender and race [90]; and discrimination against online customers based on socioeconomics [179]. Similarly, the Illinois Attorney General sent letters to six major hiring websites after users complained about age discrimination [121]. Although there are policies and best practices that employers may adopt to address biases in traditional hiring contexts, mitigating (and even detecting) these issues in online, algorithmically-driven contexts remains an open challenge [28, 107].

In this study, I examine the candidate ranking algorithms used by three of the largest hiring websites: Indeed, Monster, and CareerBuilder. My goal is to investigate gender-based inequalities in the ranking algorithms used by major *resume search engines*, which allows recruiters to proactively search for candidates based on keywords and filters. Like any search engine, these tools algorithmically rank candidates, with those at the top being more likely to be seen and clicked on by recruiters [61, 158]. However, if the ranking algorithm takes demographic features into account (explicitly or inadvertently), it may produce rankings that systematically disadvantage some candidates.

I organize my study along four questions:

CHAPTER 1. INTRODUCTION

1. Do the ranking algorithms used by major resume search engines exhibit *ranking bias* with respect to candidate’s genders? I define ranking bias as statistically significant differences in rank when comparing candidates with different genders.¹
2. Are the ranking algorithms *fair*? I define fairness as assigning adjacent ranks to candidates that have the same observable features (e.g., experience and education), regardless of gender. In other words, does rank in search results depend on gender, even when controlling for other variables?
3. If rank does depend on gender, does this mean that the ranking algorithms are using gender as a feature, or are other features serving as proxies for gender?
4. Can ranking bias and unfairness in search results be addressed using different ranking algorithms? If so, what are the tradeoffs when choosing to enforce one property or the other?

To answer these questions, I collect data from Indeed, Monster, and CareerBuilder. I ran queries on each site’s resume search engine for 35 job titles across 20 American cities between May and October 2016, and recorded the search results. My final dataset includes over 855K job candidates. Intuitively, my data corresponds to a recruiter’s perspective of these sites, including the candidates’ profiles and their rank in the search result lists.

Unfortunately, my analysis shows that the candidate ranking algorithms used by the three hiring websites exhibit both ranking bias and unfairness. To mitigate the *gender effect*, I propose two alternative ranking methods that each obey a different fairness constraint.

Ranking Bias and Unfairness. Using the Mann-Whitney U test, I find that 18.2%, 13.5%, and 18.2% job title/city pairs on Indeed, Monster, and CareerBuilder (respectively) exhibit significant ranking bias ($p < 0.05$ or smaller in all cases). For some job titles, there is significant ranking bias in 40% – 60% of the cities. In almost all cases, female candidates appear at lower ranks than male candidates.

Using a Mixed Linear Regression with $\log_2(\text{rank})$ as the dependent variable and all other visible candidate features as independent variables, I find that the gender effect is statistically significant on all three websites ($p < 0.05$ or smaller in all cases). The sign of the gender coefficient indicates that female candidates appear at lower ranks² than male candidates, even when controlling for all visible candidate features (e.g., current job title, experience, education, and skills).

¹Note that the term *bias* here is not the difference between the expected value of an estimator and the true value of the estimand; rather, I use the term to describe that the difference in rank between two groups is statistically significant.

²In this thesis, I use the terms “top” and “high” to refer to desirable ranks in search results (e.g., rank 1). This is the standard terminology used in Information Retrieval literature [61, 104].

CHAPTER 1. INTRODUCTION

Based on controlled tests using self-crafted resumes, I show that the ranking algorithms on these sites **do not explicitly use gender as a feature**; however, they do extract hidden features directly from candidates' resumes that are not visible in the search results. I hypothesize that one (or more) of these hidden features is a proxy for gender, and since I cannot control for it, this leads to the significant gender coefficient that I observe in my regressions.

Gender Effect Mitigation. I propose two alternative ranking methods that each obey a different fairness constraint. The *unbiased ranker* produces search results that do not exhibit ranking bias with respect to gender. In contrast, the *neutral ranker* increases the rank of females to compensate for the negative gender coefficient I observe in my regression models (while leaving the weight of all other features unchanged). I compare the original search results produced by Indeed, Monster, and CareerBuilder to search results that have been re-ranked according to my two methods, and show that my methods successfully remove bias and unfairness (respectively). However, I also show that my two methods lead to opposite outcomes with respect to the median rank of female candidates, the gap in rank between men and women, and the uniformity of feature values in search results.

1.4 Outline

The remainder of this thesis is organized as follows. §2 covers the related work that inspire my research. §3 presents my study of Uber's surge pricing algorithm. In §4, I explore the Buy Box and dynamic pricing on Amazon marketplaces. I examine the gender effect on ranking on hiring websites in §5. Finally, I conclude with findings and implications of my thesis in §6.

Chapter 2

Related Work

In this section, I present the related work that inspired and informed my thesis. I start with studies covering price competition and issues on online marketplaces. Then, I briefly cover studies on hiring discrimination in labor markets. Next, I present work that improves privacy and transparency on the web. Finally, I introduce algorithmic auditing and studies that applies this methodology.

2.1 Online Marketplaces

Online marketplaces, compared to traditional marketplaces, offer a platform capable of hosting an enormous number of sellers, products, customers, and transactions. The interplay of multiple parties, e.g., sellers vs. buyers and humans vs. pricing bots, paints a complex picture of *price competition* in this ecosystem. At the same time, the complexities of operating online bring *fraud and privacy* issues to online marketplaces.

2.1.1 Price Competition

With easy access to the internet and computation technologies, e-commerce sellers are able to adjust their prices automatically by setting algorithmic rules against other competitors in the market. These sellers are playing a *pricing game* in the marketplace. [22, 32, 139] model a pricing game played by the sellers and study the properties of its equilibria as a function of the dependencies among goods/services offered by the sellers. [21, 49, 50] extend the traditional price competition model proposed by Bertrand [32] to combinatorial settings. Babaioff et al. model price competition in marketplaces where equilibria rarely exist [23].

CHAPTER 2. RELATED WORK

Researchers have also empirically analyzed price competition on online marketplaces. Chevalier et al. measure 18,000 books to estimate price indices and the amount of price competition at Amazon.com and BarnesandNoble.com [55]. In another Amazon.com study [193], Zhu et al. analyze the competition between Amazon.com and the third-party sellers on its platform. In the study, the authors find that Amazon’s entry to compete with third-party sellers is positively correlated with the popularity and customer ratings of third-party sellers’ products. In addition, they also show that Amazon’s competition reduces the shipping costs of affected products, and hence increases their demand. However, Amazon’s entry into competition also depresses the business growth for small third-party sellers. Lastly, [25] analyzes the impact of “snipers”, who place bids in the final seconds in an online auction, on user behavior on eBay.

2.1.2 Fraud and Privacy

Online marketplaces bring customers convenience, low prices, and a vast inventory of products. However, the anonymous nature of online marketplaces makes them vulnerable to manipulation and fraud conducted by unscrupulous parties. [118, 137, 147] study insincere sellers that generate opinion spam and artificial ratings to manipulate the reputation systems on online marketplace. Xu et al. conduct an empirical analysis on the Seller Reputation Escalation (SRE) ecosystem that provides a skill-purchasing service for escalating business’ reputations on the Taobao online marketplace [190].

Several empirical studies have also shown that online marketplaces can cause privacy issues for consumers. Minkus et al. reveal that attackers can correlate the highly sensitive user information from public profiles in eBay’s feedback system with their social network profiles on Facebook [131]. Similarly, [153] discovers leakage of personal information and detailed shopping habits from online merchants to payment providers (e.g., PayPal).

2.2 Hiring Discrimination

Labor discrimination is a long standing, troubling aspect of society. Discrimination may impact workers’ wages or opportunities for advancement, but in this thesis I specifically focus on *hiring discrimination*, which occurs when discrimination impacts the candidates that are selected to fill open positions. Recent literature demonstrates that hiring discrimination still impacts the modern job market [149], and may be based on gender, race [30, 182], sexual orientation [188], disability [77], or age [30, 73].

CHAPTER 2. RELATED WORK

Kuhn et al. used data from a Chinese online job board to examine *explicit* gender discrimination in hiring [113]. Gender discrimination in hiring is not illegal in China, which gave the authors a window into firm’s behavior in an unconstrained market. In contrast, gender discrimination is illegal in the U.S., making it more difficult to study because individuals and firms attempt to hide this behavior.

One of the key tools used to study hiring discrimination in countries where the practice is illegal is the *audit* or *correspondence study*. In this methodology, the researchers probe the hiring practices of a target by submitting carefully crafted resumes, or by sending actual human participants in for interviews [30, 33, 157]. By carefully constructing the treatments to only differ by specific demographic features (e.g., gender), the researchers can measure the degree to which these variables correlate with hiring outcomes [148].

Scholars and regulators have begun to focus on the ways that big data and algorithms can create hiring discrimination. Pauline T. Kim thoroughly catalogs how data-driven systems that evaluate job seekers may introduce new forms of bias against members of protected classes [107]. One potential driver of algorithmic bias identified by Kim and others [28] occurs when subpopulations are not well-represented in the data used to train an algorithm. This issue is not hypothetical: in 2017, the Illinois Attorney General launched an investigation against six major online job boards (including Indeed, Monster, and CareerBuilder) after receiving complaints that the design of these websites excluded older job seekers, thus fostering age discrimination [121].

2.2.1 The Importance of Rank

In §5, I examine three hiring websites that present job seekers in ranked lists in response to queries from recruiters. Ranked lists are a common user interface design pattern across many Information Retrieval (i.e., search) systems. Numerous empirical studies have shown that the items at the top of ranked lists are more likely to be clicked by users [61, 158], and thus there is an entire Search Engine Optimization industry that aims to increase their clients’ rank within critical systems like Google Search [124, 142, 186, 187].

Presenting human job seekers in a ranked list can be framed as a form of *status differentiation*, with top ranks conferring higher status. In offline contexts, status differentiation is known to be a primary cause of social inequality. For example, studies have shown that men are perceived to have higher status than women [38, 97, 136, 159, 184], while white skinned people have higher status than darker skinned people [59, 173].

CHAPTER 2. RELATED WORK

With respect to online job search, if a ranking algorithm systematically elevates candidates with specific demographic attributes, this may recreate real-world social inequality in an online context. Unfortunately, examples of this are emerging: Hannák et al. found negative correlations between race and rank on the gig-economy website TaskRabbit, even after controlling for all other worker-related features [90].

Another frame for considering the importance of ranking algorithms in hiring is that of *automation bias*. Automation bias is a cognitive bias where humans tend to believe the output of an automated decision aid even when the output is contradicted by other available evidence [62]. In my context, I can view the ranking algorithm on a hiring website as a decision aid designed to guide a recruiter towards relevant candidates. Hiring websites promote the sophistication of their search tools [42, 134], and access to them often costs money [46, 135], which may increase recruiters' perception that the top candidates returned by these tools are truly "the best" on the market. Naïve acceptance of search results may dissuade recruiters from investigating and contacting lower-ranked, but possibly equally qualified candidates.

2.3 Auditing Algorithms

As sophisticated algorithms automate more facets of daily life, researchers have begun to investigate whether these opaque algorithms may (inadvertently) cause harm to users. The process of examining a black-box system has become known as an *algorithm audit*, as the methodology draws inspiration from classic audit studies [162]. There is a growing body of literature that uses algorithm audits to understand the systems that impact people's daily lives.

2.3.1 Filter Bubbles

Personalization is a technique that tailors content per individual based on the user's personal information [151]. While personalization may provide more relevant content to end users, the technique has also drawn criticism because it may hide information with different viewpoints, and isolate users into individual cultural or ideological bubbles [37, 71, 94, 98]. This controversial phenomenon is known as the *Filter Bubble* of personalization [150].

Researchers are studying the prevalence and impact of Filter Bubbles on search engines. Hannak et al. develop a methodology to measure the personalization in web search results and identify the features that caused the personalization [88]. Kliman-Silver et al. explore the impact of

CHAPTER 2. RELATED WORK

location-based personalization on Google Search results [110]. To detect and catalog personalization of international borders in online maps, Soeller et al. build a system to monitor controversial international borders on Google and Bing [169].

Personalization is also present on online social networks. For example, Facebook’s News Feed algorithm decides the posts each user sees when using the service [24, 145]. To understand users’ perceptions of the algorithmic curation in Facebook’s News Feed, Rader et al. analyze open-ended survey responses from users and identify several patterns of beliefs and their implications [155]. Eslami et al. build a tool to present Facebook users with algorithmically curated and uncured news feeds, and use the tool to study users’ perceptions about the algorithm [72]. To measure users’ exposure to cross-cutting content, Bakshy et al. examine 10.1 million U.S. Facebook users and their interactions with socially shared news, and argue that homophily has a stronger effect than the News Feed curation algorithm on limiting the visibility of political messages on Facebook [27].

2.3.2 Price Discrimination

Automated algorithms are becoming increasingly ubiquitous on online marketplaces. However, the impact of these algorithms on users are often poorly understood, and are not always positive for users. Price discrimination, algorithmically setting the product price for a user based the user’s personal information, has received extensive scrutiny by academic researchers. Mikians et al. empirically demonstrate the existence of both price and search discrimination by tracking 200 online vendors for 20 days, and uncover the information vectors used to facilitate the discrimination [129]. From the same research group, Mikians et al. conduct a follow-up study by implementing a browser extension to crowd-source the data collection from real users, and find the discrepancy in product prices can reach 200% [130]. Similarly, in [58], Clemons et al. examine the presence of price dispersion and product differentiation on the airline ticket offerings from online travel agents, and find that tickets prices vary by as much as 18% across agents. Lastly, Hannak et al. develop a methodology for accurately measuring when price steering and discrimination occur, implement it for a variety of e-commerce web sites, and investigate the effect of user behaviors on personalization [89].

2.3.3 Gender and Racial Discrimination

Gender and racial discrimination is extensively studied in real-world society [48, 154, 164, 181]. Unfortunately, these issues are also present on online services. Latanya Sweeney investigated racial discrimination on Google AdSense, and discovered that the advertising system was more likely

CHAPTER 2. RELATED WORK

to deliver criminal background check advertisements when searching for African American sounding names than white names [176]. Similarly, Datta et al. developed an automated tool that explored how user behaviors, Google’s ads, and Ad Preferences interact, and found that females were shown fewer advertisements for high-paying jobs [64]. Two studies focused on eBay marketplace uncovered that female and African American sellers sold fewer products, and at lower prices, than equivalent male caucasian sellers [20, 112].

In the sharing economy, Edelman et al. found that rental applications on Airbnb from guests with distinctively African American names were 16% less likely to be accepted relative to identical guests with distinctively white names [69]. Similar unequal treatments have been observed on Uber and Lyft. Ge et al. found that ride requests from passengers with African American sounding names were more likely to be canceled, while female passenger were more likely to have longer, more expensive rides [75]. Finally, on online freelance marketplaces like TaskRabbit, researchers have found that workers accept fewer job requests from disadvantaged or suburban areas [180], and that gender and race are significantly correlated with worker evaluations, which could harm the employment opportunities afforded to the workers [91].

A recent study by the hiring website HIRED analyzed the wage gap among employees from different demographic backgrounds in the tech industry [109]. Unfortunately, the analysis revealed that female, non-white, and LGBTQ employees face extensive salary disadvantages compared to white men.

2.3.4 Privacy and Transparency

Researchers have begun to develop tools to make online data collection by companies more transparent, and help preserve user privacy. To increase transparency, Lécuyer et al. develop a scalable personal data tracking system that monitors the user’s data transfers and detects possible leaks of sensitive information [116]. Roesner et al. analyze the first- and third-party tracking of user information online and evaluate several defense mechanisms against web tracking [160]. In a recent study, Ren et al. implement a cross-platform system that reveals personally identifiable information (PII) leaks, and gives users control over them without requiring any special privileges or custom OSes [156]. [29, 83] measure the information flows in online advertising networks and provide insights to build systems offering more user control and transparency.

Chapter 3

Uber’s Surge Pricing Algorithm

There are three high-level goals of this study. *First*, I aim to understand the overall dynamics of the Uber service. *Second*, I want to determine how Uber calculates surge multipliers. *Third*, I want to understand whether surge pricing is effective at mitigating supply/demand imbalances.

To answer these questions, I begin in §3.1 by introducing Uber, its client-side and driver-side app, and surge pricing. In §3.2, I present my approach to collect data from Uber, and the methodology to measure key variables such as *supply*, *demand*, *surge multiplier*, etc., to conduct my research. I also validate my data collection and measurement methodology in this section. In §3.3, I characterize temporal and spatial dynamics of supply and demand on Uber, and analyze Uber’s surge pricing algorithm in detail in §3.4. Based on these findings, I present a strategy for avoiding surge prices in §3.5. Finally, I summarize key findings in §3.6.

3.1 Background

In this section, I briefly introduce Uber, surge pricing, and technical details about the Uber service.

Uber. Founded in 2009, Uber is a “ride sharing” service that connects people who need transportation with a crowd-sourced pool of drivers. Unlike typical transportation providers, Uber does not own any vehicles or directly hire drivers. Instead, Uber drivers are independent contractors (known as “partners”) who use their own vehicles and set their own schedules. As of May 2015, Uber is available in 200 cities in 57 countries¹, and claims to have 160K active drivers in the U.S.

¹<https://www.uber.com/cities>

CHAPTER 3. UBER'S SURGE PRICING ALGORITHM

alone [86].

Uber provides a platform that connects passengers to drivers in real-time. Drivers use the Uber *Partner app* on their smartphone to indicate their willingness to accept fares. Passengers use the Uber *Client app* to determine the availability of rides and get estimated prices. Uber's system routes passenger requests to the nearest driver, and automatically charges passengers' credit cards at the conclusion of each trip. Uber retains 20% of each fare and pays the rest to drivers.

Depending on location, Uber offers a variety of different services. UberX and UberXL are basic sedans and SUVs that compete with traditional taxis, while UberBLACK and UberSUV are luxury vehicles that compete with limousines. UberFAMILY is a subset of UberX and UberXL cars equipped with car seats, while UberWAV refers to wheelchair accessible vehicles. UberT allows users to request traditional taxis from within the Uber app. UberPOOL allows passengers to save money via carpooling, i.e., Uber will assign multiple passengers to each vehicle. UberRUSH is a delivery service where Uber drivers agree to courier packages on behalf of customers.

Surge Pricing. Uber's fare calculation changes depending on local transportation laws. It typically incorporates a minimum base fare, cost per mile, cost per minute, and fees, tolls, and taxes. The base fare and distance/time charges vary depending on the type of vehicle (e.g., UberX vs. UberBLACK).

In 2012, Uber introduced a dynamic component to pricing known as the *surge multiplier*. As the name suggests, fare prices are multiplied by the surge multiplier; typically the multiplier is 1, but during times of high demand it increases. Uber has stated that there are two goals of this system: *first*, higher profits may increase supply by incentivizing drivers to come online. *Second*, higher prices may reduce demand by discouraging price-elastic customers.

Little is known about Uber's surge price algorithm, or whether the system as a whole is successful at addressing supply/demand imbalances. As shown in Figure 3.1, surge multipliers vary based on location, and recent measurements suggest that Uber updates the multipliers every 3-5 minutes [67]. Uber has a patent pending on the surge price algorithm, which states that features such as car supply, passenger demand, weather, and road traffic may be used in the calculation [141].

Passenger's Perspective. Uber's apps provide different information to passengers and drivers. The Client app displays a map with the eight closest cars to the user (based on the smartphone's geolocation), and the Estimated Wait Time (EWT) for a car. The app provides separate eight-car inventories and EWTs for each type of Uber. Users are not shown the surge multiplier until they attempt to request a car (and only if it is >1). Although the app initially assumes that the user's

CHAPTER 3. UBER’S SURGE PRICING ALGORITHM

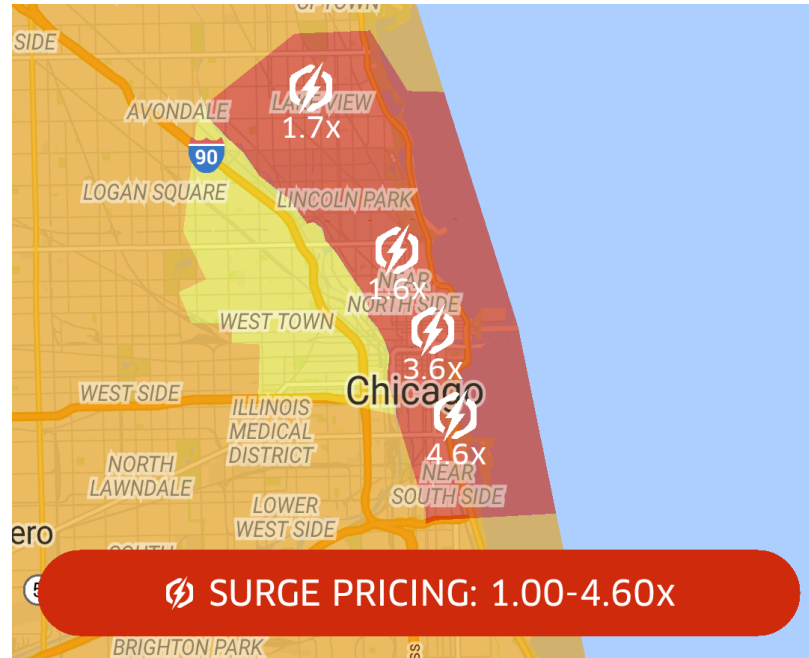


Figure 3.1: Screenshot of the Uber Partner app.

current location is their pickup point, the user may move the map to choose a different pickup point. This new location may have a different EWT and/or surge multiplier than the original location.

Driver’s Perspective. In contrast to the Client app, the Partner app displays very different information to drivers. As shown in Figure 3.1, the centerpiece of the Partner app is a map with colored polygons indicating areas of surge. Unlike the Client app, the locations of other cars are not shown. In theory, this map allows drivers to locate areas of high demand where they can earn more money. In practice, drivers often use the Partner and Client apps concurrently, to see the exact locations of competing drivers [19]. The Partner app’s map also suggests that Uber calculates discreet surge multipliers for different geographic areas. I empirically derive these *surge areas* in § 3.4.

3.2 Methodology

In this section, I discuss my approach for collecting data from Uber. I begin by motivating San Francisco (SF) and Manhattan as the regions for my study. Next, I discuss my methodology for collecting data from the Uber Client app, and how I calibrated my measurement apparatus. Finally, I validate my methodology using simulations driven by ground-truth data on all taxi rides in NYC in

2013.

3.2.1 Selecting Locations

In this study, I focus on the dynamics of Uber in SF and Manhattan. As I discuss in §3.2.2 and §3.2.3, there are practical issues that force me to constrain my data collection to relatively small geographic regions. Furthermore, not all regions are viable or interesting: Uber does not offer service everywhere, and many places will have few cars and passengers (i.e., rural areas).

I chose to focus on SF and Manhattan for four reasons. *First*, San Francisco and New York City have the 2nd and 3rd largest populations of Uber drivers in the U.S. (Los Angeles has the largest population) [86]. *Second*, SF was Uber’s launch city, and recent measurements suggest that it accounted for 71% of all “taxi” rides in the city in 2014 (the highest percentage of any U.S. city; Uber accounted for 29% of all rides in NYC during 2014) [174]. *Third*, SF and NYC are very different cities in terms of culture and access to public transportation, which may lead to interesting differences in the dynamics of Uber.

Fourth and finally, Manhattan is a useful location to measure Uber because there also exists a publicly available dataset of all taxi rides in NYC for 2013 [132]. I leverage this data in §3.2.5 to validate the accuracy of my Uber measurement methodology.

3.2.2 The Uber API

Now that I have chosen locations, the next step is to collect data from Uber in these two areas. Like many modern web services, Uber provides an HTTP-based API for third-party developers to retrieve information about the state of the service. In my case, the `estimates/price` and `estimates/time` endpoints are most useful. The former takes longitude and latitude as input, and returns a JSON-encoded list of price estimates (including surge multipliers) for all car types available at the given location. The latter is similar, except it returns EWTs. Uber imposes a rate limit of 1,000 API requests per hour per user account.

While data from the Uber API is useful, it is not sufficient for this study, since it does not include information about car supply or passenger demand. Thus, I only rely on API data for specific experiments in §3.4.

3.2.3 Collecting Data from the Uber App

To overcome the shortcomings of the Uber API, I leverage the Uber Client app. After a user opens the app and authenticates with Uber, the app sends `pingClient` messages to Uber’s server every 5 seconds. Each ping includes the user’s geolocation, and the server responds with a JSON-encoded list of information about all available car types at the user’s location. For each car type, the nearest eight cars, EWT, and surge multiplier are given. Each car is represented by a unique ID, its current geolocation, and a path vector that traces the recent movements of the car.

To gather this data, I wrote a script that emulates the exact behavior of the Client app. My script logs-in to Uber, sends `pingClient` messages every 5 seconds, and records the responses. By controlling the latitude and longitude sent by the script, I can collect data from arbitrary locations. I created 43 Uber accounts (each account requires a credit card to create), giving me the ability to “blanket” a small geographic area with measurement points. To simplify my discussion, I refer to these 43 measurement points as “clients”.

While I was collecting data I never encountered rate limits or had my accounts banned. This indicates that I very likely was not detected by Uber. Although it is possible that Uber detected my clients and fed them false data, it is much more plausible that Uber would have simply banned my clients if they were concerned about my measurements.

Measuring Supply and Demand. Using the data returned by `pingClient`, I can approximate the aggregate supply and demand within my measurement region. To measure *supply*, I can simply count the total number of unique cars observed across all measurement points; each of these cars represents a driver who is looking to provide a ride. To measure *demand*, I can measure the aggregate number of cars that go *offline* (disappear) between responses; one of the reasons a car may go offline is because it picked up a rider (I discuss other potential reasons, and how I handle them, below).

Limitations. Although `pingClient` returns more information than the Uber API, there are still four limitations that I must address. *First*, clients only receive information about the eight closest cars. Thus, to measure the overall supply of vehicles in a geographic area, I must position the 43 clients such that they completely cover the area. This situation is further complicated by the fact that each client’s visibility changes as the density of Uber cars fluctuates (e.g., cars are dense during rush hour but sparse at 4am). In §3.2.4, I perform calibration experiments to determine the appropriate distance to space my clients.

Second, the demand I am able to estimate from my data is *fulfilled demand*, i.e., the number

CHAPTER 3. UBER’S SURGE PRICING ALGORITHM

of cars that pick up passengers. Uber does not provide public data about *quantity demanded*, i.e., the number of passengers that request rides. The difference between fulfilled and quantity demand is that some passengers may request a ride but not receive one due to supply shortages. Thus, in this study, when I refer to “demand”, I am talking about fulfilled demand.

Third, my measurement of demand may overestimate the true demand because there are three reasons why a car might disappear between one response and the next: 1) the car drives outside my measurement area, 2) the driver accepts a ride request, or 3) the driver goes offline. I can disambiguate case 1 since the Client data includes the path vector for each car. Although I cannot disambiguate cases 2 and 3, I can still use car disappearances as an upper-bound on the fulfilled demand within the measurement area.

Fourth, data from the Client app does not allow me to track individual Uber drivers over time. Although each car is assigned a unique ID, these IDs are randomized each time a car comes online. Unfortunately, there is no way to overcome this limitation, and thus none of my experiments rely on tracking individual drivers.

Phantom Cars. Several press articles claim that Uber’s Client app does not display data about actual Uber cars; instead, they claim that the cars are “phantoms” designed to give customers the illusion of supply [161]. Uber has publicly disputed these claims [57, 189], explaining that the data shown in the Client app is as accurate as possible, given practical constraints like the accuracy of smartphone GPS measurements. Furthermore, Uber stated that car locations may be slightly perturbed to protect drivers’ safety. I have not observed any evidence in my data to suggest that the cars are phantoms; on the contrary, the cars in my data exhibit all the hallmarks of human activity, such as diurnal activity patterns (see §3.3). If Uber does present phantom cars, it is likely that they only do so in rural areas with low supply, rather than in major cities like Manhattan and SF.

Uber Driver App. As shown in Figure 3.1, the Driver app also includes useful information (i.e., the surge map). However, only registered Uber drivers may log in to the Driver app. I attempted to sign-up as an Uber driver, but unfortunately Uber requires that drivers sign a document prohibiting data-collection from the Driver app. I opted not to sign this agreement. Instead, in §3.4, I reconstruct the surge map based on data from the Uber API.

Ethics. While conducting this study, I was careful to collect data in an ethical manner. *First*, I do not collect any personal information about any Uber users or drivers. I discussed my study with the Chair of my University’s Institutional Review Board (IRB); she evaluated it as not being subject to IRB review because I did not collect personal information or impact any user’s

CHAPTER 3. UBER’S SURGE PRICING ALGORITHM

environment.

Second, I minimize my impact on Uber users and drivers. Before I began my data collection, I conducted an experiment to see if my measurements would impact Uber users by artificially raising the surge price. Fully discussed below, my results strongly suggests that my measurements have no impact on the surge multiplier. Moreover, at no point in this study did I actually request rides from any Uber driver, and drivers are not able to observe my measurement clients in the Driver app.

Third, I minimized my impact on Uber itself by collecting just the data I need to perform the study. The overall effect of my measurements was the same as 43 extra users running the Uber Client app. Given that Uber claims millions of users worldwide, I believe this is a worthwhile tradeoff in order to conduct this research.

Other Ride-Sharing Services. Although I attempted to collect data from other ride sharing services, these efforts were not successful. Lyft implements “prime time” pricing, but this data is only available *after* a user requests a ride. Thus, there was no ethical way for me to collect this data. Sidecar does not implement surge pricing; instead, drivers set their own rates based on time and distance. These additional variables make it difficult to systematically collect price information.

3.2.4 Calibration

The next step in my methodology is determining the locations for my 43 clients in SF and Manhattan. This step is crucial; on one hand, if I distribute the clients sparsely, I may only observe a subset of cars and thus underestimate supply and demand (recall that `pingClient` responses from Uber only contain the closest eight cars to each client). On the other hand, if the clients are too close together, the cars they observe will overlap, and I will fail to observe supply and demand over a sufficiently large geographic area.

To determine the appropriate placement of my 43 clients, I conducted a series of experiments between December 2013 and February 2014. In my first experiment, I chose a random location in Manhattan and placed all 43 clients there for one hour. I then repeated this test over several days with different random locations around Manhattan and SF. The results of these experiments reveal two important details about Uber: *first*, during each test, all 43 clients observed exactly the same vehicles, surge multipliers, and EWTs. This strongly suggests that the data received from `pingClient` is deterministic.

Second, when the clients were placed in areas where I would not expect to see surge (e.g., residential neighborhoods at 4 a.m.), all 43 clients recorded surge multipliers of 1 for the entire hour.

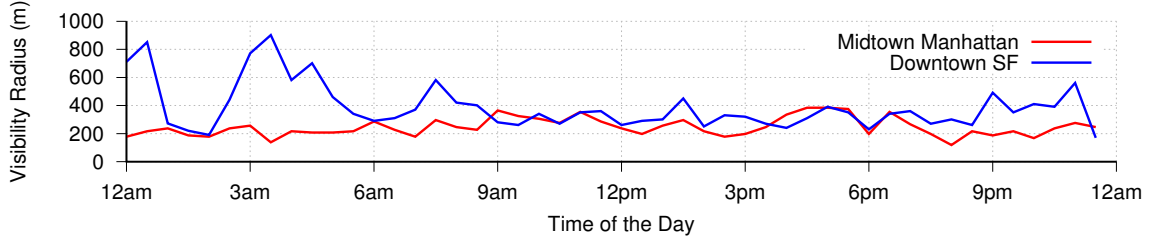


Figure 3.2: Visibility radius of clients in two cities.

This strongly suggests that my measurement methodology does not *induce* surges. As I show in Figure 3.8, fulfilled demand in midtown Manhattan peaks around 100 rides per hour, so 43 clients is a significant enough number that I would expect surge to increase if the algorithm took “views” into account.

The goal of my next experiment is to measure the *visibility radius* of clients. Intuitively, this is the distance from a client to the furthest of the eight cars returned by `pingClient`. Once I know the visibility radius in SF and Manhattan, I can determine the placement of my 43 clients.

To calculate the visibility radius, I conduct the following experiment. I 1) place 4 clients, denoted as $C = \{c_1, c_2, c_3, c_4\}$, at the same geolocation; 2) each of the clients “walks” 20 meters Northeast, Northwest, Southeast and Southwest (respectively) every 5 seconds; 3) the experiment halts when $|\bigcap_c V_c| = 0$, where V_{c_i} is the set of cars observed by client c_i ; 4) record the distance D_c from each $c \in C$ to the starting point. Given this information, I calculate the visibility radius r (consider a 45° - 45° - 90° triangle where r is the leg and D_c is the hypotenuse) as:

$$r = \frac{1}{4} \sum_c \frac{D_c}{\sqrt{2}} \approx 0.1768 \times \sum_c D_c$$

Figure 3.2 shows the measured radii in meters when the clients were placed in downtown SF and midtown Manhattan. I chose these specific locations because they are the “hearts” of these cities, and thus they are likely to have the highest densities of Uber cars. As expected, the visibility radius changes throughout the day, with the most obvious difference being night/day in SF. There are also differences between the cities: the average radius is 247 ± 2.6 meters in Manhattan, versus 387 ± 6.8 meters in SF.²

In the end, I chose 200 meters as the radius for my data collection in midtown Manhattan, and 350 meters in downtown SF. These values represent a conscientious trade-off between obtaining

²Throughout this study, I present the 95% confidence interval (CI) of the mean value.

complete coverage of supply/demand and covering a large overall geographic area. Figures 3.3a and 3.3b depict the exact positions where I placed my clients in SF and Manhattan.³

3.2.5 Validation

My final step is to validate my measurement methodology. The fundamental challenge is that I do not have ground-truth information about supply and demand on Uber; I attempt to mitigate this through careful placement of my clients, but the key challenge is having confidence that I will observe the vast majority of cars.

To address this issue, I constructed an Uber simulator powered by ground-truth data on NYC taxis [132]. The NYC taxi data includes timestamped, geolocated pickup and dropoff points for all taxi rides in NYC in 2013. Each taxi is assigned a unique ID, so its location can be tracked over time. My simulator takes the taxi data as input, and plays the rides back in real-time. Since the taxi data only includes pickup and dropoff points, the simulator “drives” each taxi in a straight-line from point-to-point. I assume that a taxi has gone “offline” if it is idle for more than 3 hours (this filter only removes 5% of taxi sessions in the data).

I built an API in my simulator that offers the same functionality as Uber’s `pingClient`: it returns the eight closest taxis to a given geolocation. Just as with Uber, the ID for each taxi is randomized each time it becomes available. Given this API, I used my methodology from §3.2.3 and §3.2.4 to measure the supply and demand of taxis over time. If the measured values from the simulator’s API are similar to the ground-truth values, I can confidently say that my methodology will also collect accurate data from Uber.

Calibration. To make my simulation fair, I calibrated it by using four taxi clients to determine the visibility radius for taxis in midtown NYC (see §3.2.4). Taxis are much denser than

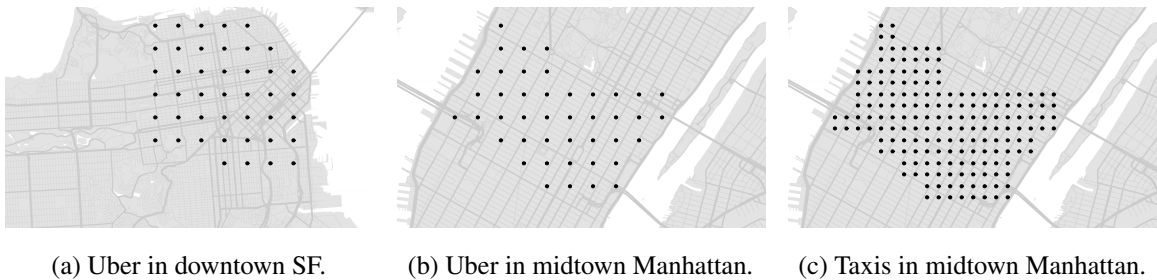


Figure 3.3: Locations of my Uber and taxi measurement points in SF and Manhattan.

³All map images used in this thesis are ©2015 Google.

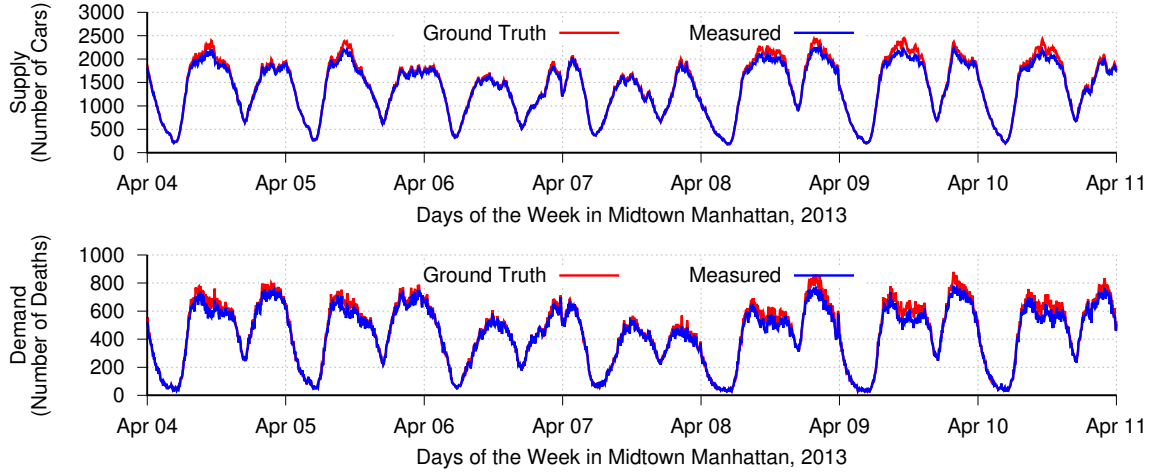


Figure 3.4: Measured and ground-truth supply (left) and demand (right) of taxis in midtown Manhattan.

Ubers in this area, so $r = 100$ meters is commensurately smaller. Figure 3.3c shows the locations of my 172 taxi clients; compared to Uber clients, it takes 300% more taxi clients to cover midtown.

Results. Using my taxi clients, I measured the supply and demand of taxis in the simulator between April 4–11, 2013. I chose these dates because they correspond to the same month and week of my Uber measurements (except in 2013 versus 2015, see §3.3).

As I discuss in §3.2.3, neither Uber nor my simulator return direct information about demand. Instead, I assume that cars that 1) disappear from the measured data, and 2) were not driving near the outer edge of the measurement polygon were booked by a passenger.⁴ I refer to these events as *deaths*.

Figure 3.4 plots the measured and ground-truth taxi supply and demand per 5-minute interval. The two lines are almost indistinguishable since my taxi clients capture 97% of cars and 95% of deaths. The results provide strong evidence that my measurement methodology captures most of Uber’s supply and demand.

⁴Restriction (2) is conservative: cars near the edge of the measurement area may disappear because they were booked, or because they drove outside the measurement polygon.

3.3 Analysis

In this section, I analyze supply, demand, and wait times on Uber. I begin by briefly introducing my datasets and how I cleaned them. Next, I examine how the dynamics of Uber change over time and spatially across cities. I focus the majority of my analysis on UberX, since (as I will show) they are the most common type of Uber by a large margin. I defer detailed analysis of surge multipliers and the surge pricing algorithm to §3.4.

3.3.1 Data Collection and Cleaning

For this study, I collected data from 43 clients placed in midtown Manhattan and downtown SF between April 3rd–17th (391 GB of data containing 9.3M samples) and April 18th–May 2nd (605 GB of data, 9.4M samples), respectively. Data was collected using the Uber client methodology described in §3.2.3. The locations of my clients are shown in Figures 3.3a and 3.3b. I also collected more limited datasets from the Uber API, as well as client data between December 27th, 2014–March 1st, 2015; I use this data in §3.4.

Before proceeding with analysis, my data must be cleaned to remove outliers. Figure 3.5 shows the *lifespan* of UberX cars in my dataset, measured as the time delta during which I observe a car’s ID. Problematically, ~50% of UberXs have a lifespan of zero, i.e., I only observe them in a single `pingClient` response. I refer to these cars as being *short-lived*, and observe similar trends for other types of Ubers.

The obvious question is: what is the cause of short-lived cars? Fortunately, I discovered that short-lived cars are an easily understood artifact of my measurement methodology. I examined the GPS coordinates of all short-lived cars, and found that 80% of them are *outsiders*: they are $> r$ meters away from the nearest client in my measurement polygon. Figure 3.6 plots the distances of short-lived *insiders* (those within the measurement polygon), and shows that 100% of them are $< r$ meters from the polygon boundary.

These results reveal that short-lived cars are due to the fact that `pingClient` only returns information about the eight nearest cars. Cars that are outside the measurement polygon, or $< r$ meters from its edge, may only be observed by a single client. These cars may get pushed out of `pingClient` responses by closer vehicles, or the car may only briefly be near my measurement area. In contrast, cars well-within the boundaries can potentially be observed by many clients, even as they drive around. Thus, I can safely filter short-lived cars from my dataset, and focus in the remainder of the thesis only on cars that are driving within the bounds of my measurement area.

CHAPTER 3. UBER'S SURGE PRICING ALGORITHM

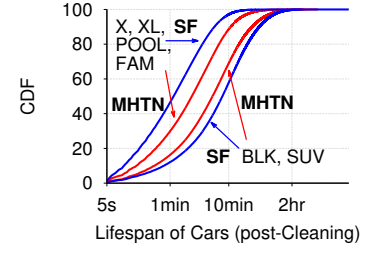
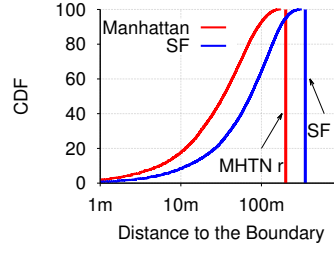
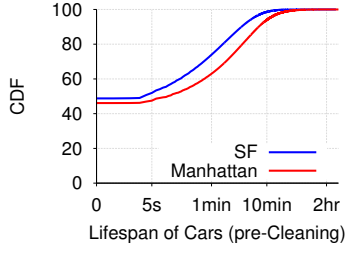


Figure 3.5: Lifespan of UberX cars in Manhattan and SF before data cleaning. Figure 3.6: Distance of short-lived insiders to the boundary. Figure 3.7: Lifespan of different types of Uber cars after data cleaning.

Car Lifespan. Figure 3.7 shows the lifespans of Ubers after removing short-lived cars. I observe similar trends in Manhattan and SF: $\sim 90\%$ of low-priced Ubers (X, XL, FAMILY, and POOL) live for <10 minutes, while $\sim 65\%$ of high-priced Ubers (BLACK and SUV) live for <10 minutes. There are two possible reasons for this observation: first, demand for low-priced Ubers may be higher than for expensive Ubers. Second, as I show in §3.3.2, low-priced Ubers greatly outnumber high-priced Ubers, so there may simply be more churn amongst the former due to their greater prevalence.

3.3.2 Dynamics Over Time

Next, I examine how supply, demand, surge multiplier, and EWT vary over time. I calculate supply by counting all the unique car IDs observed by my 43 clients during each 5-minute interval. Demand is the number of cars that disappear during each 5-minute interval (excluding cars that drive outside the measurement area). Surge multiplier and EWT are averaged across each 5-minute interval and all 43 clients. I explain why I chose 5-minute intervals in §3.4.

Note that my measurements for supply and demand are approximations. As shown in §3.2.5, my methodology may miss some cars, leading me to slightly underestimate supply. Similarly, I cannot disambiguate cars that pickup a passenger from cars that simply go offline. Thus, my demand numbers are an upper-bound. Finally, a driver that repeatedly goes online and offline during a 5-minute interval will appear as multiple unique cars to me, since cars are assigned a fresh ID each time they come online.

Unsurprisingly, Figure 3.8 shows that Uber exhibits regular daily trends: all four quantities peak during the day and decline at night. I also observe that supply, demand, and EWT have local

CHAPTER 3. UBER'S SURGE PRICING ALGORITHM

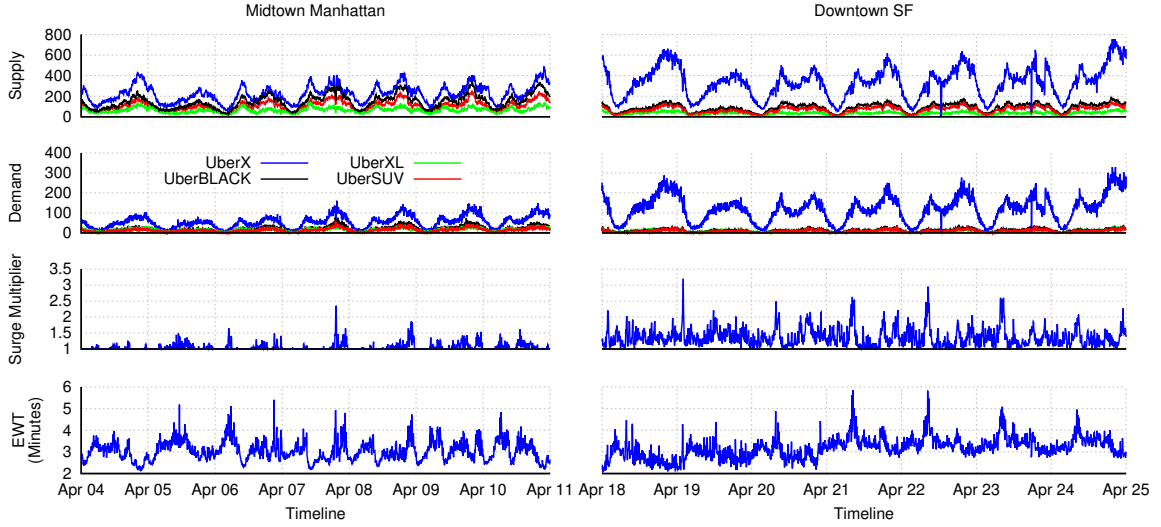


Figure 3.8: Supply, demand, surge multiplier, and EWT over time for midtown Manhattan and downtown SF. Surge multiplier and estimated wait time (EWT) are only shown for UberX. Diurnal patterns are observed in supply and demand, but the characteristics of the surge multiplier show less predictability.

peaks during morning and afternoon rush hour.⁵ Rush hour peaks are less prominent for surge pricing, and I show in §3.4 that surge pricing is extremely noisy.

Both cities exhibit the same rank ordering of Uber types. UberXs are most prevalent, followed by UberBLACK, UberSUV, and UberXL. Both cities also have other types of Ubers (e.g., UberRUSH, UberFAMILY, *etc.*), however there are only 4 cars of these types on the road on average. Manhattan does have a significant number of UberT's, but these are not interesting in my context since they are not subject to surge pricing (recall that UberT corresponds to an ordinary taxi). Comparing the NYC taxi data in Figure 3.4 to Figure 3.8 I see that there are an order of magnitude more taxis in midtown Manhattan than all Ubers combined.

Despite their similarities, Figure 3.8 also reveals differences between Manhattan and SF. In my measurement region, SF has 58% more Ubers overall than Manhattan, mostly due to the large amount of UberXs in SF. The relative dearth of Ubers in Manhattan may be due to greater availability of taxis and better public transport. However, Manhattan has more UberXLs, UberBLACKs, and UberSUVs than SF.

Manhattan and SF also exhibit different surge pricing characteristics. As shown in Fig-

⁵Note that the y -axes of supply and demand have different scales.

CHAPTER 3. UBER’S SURGE PRICING ALGORITHM

ure 3.8, downtown SF *surges* (i.e., surge >1) more frequently than midtown Manhattan. Surge multipliers also tend to be higher in SF: $1.36 \pm 1 \times 10^{-4}$ on average versus $1.07 \pm 7 \times 10^{-5}$ in Manhattan. In midtown Manhattan, surge tends to increase starting at 3pm through evening rush hour Monday–Thursday. On the weekends, surge tends to peak between noon and 3pm, likely due to the influx of tourists. In SF, surge peaks at around 2.0 during morning rush hour (6am–9am) Monday–Friday. Surge also has a localized peak at 2am every night (but especially on weekends, reaching up to 3.0), which is “last call” throughout California.

Although Figure 3.8 focuses on surge multipliers of UberX, other types of Ubers exhibit similar trends. A recent report estimated that Uber accounts for 71% of “taxi” rides in SF versus 29% in NYC [174], so it is possible that this difference in demand explains the differences in surge characteristics. I defer in-depth discussion of surge pricing to §3.4.

I also observe that Uber offers expedient service in both cities. Average EWT for an UberX in midtown Manhattan is $3.0 \pm 2 \times 10^{-4}$ minutes, and $3.1 \pm 2 \times 10^{-4}$ minutes in downtown SF. In both cities, average EWT never exceeds six minutes. I observe similar trends for other types of Ubers, although rarer types (e.g., UberFAMILY) typically have slightly longer EWTs.

3.3.3 Spatial Dynamics and EWT

Next, I investigate the spatial dynamics of Uber. Figures 3.9(a) and 3.10(a) show the average number of unique UberX IDs seen per day by each of my 43 clients.⁶ Since each square in these figures is an average, each one has a different confidence interval; the min and max CI for cars in Manhattan are ± 103 and ± 205 cars, respectively. The min and max CI for cars SF is ± 170 and ± 250 . As one might expect, the distribution of cars in both cities is skewed towards commercial and tourist locations. In Manhattan, UberXs congregate between Times Square and 5th Avenue. In SF, UberXs are densest in Russian Hill, Telegraph Hill, the Embarcadero, and the Financial District (upper-right corner of Figure 3.10(a)), as well as around the University of California, San Francisco (UCSF, lower-left corner).

In contrast, Figures 3.9(b) and 3.10(b) depict the distribution of EWTs across space. Each square is the average EWT for UberXs measured every five seconds over two weeks. The min and max CI for Manhattan are $\pm 6 \times 10^{-6}$ and $\pm 2 \times 10^{-4}$ minutes, respectively; the min and max CI for SF are $\pm 7 \times 10^{-7}$ and $\pm 4 \times 10^{-5}$. I observe that there is a complex relationship between

⁶Note that these numbers are strict upper-bounds on the true number of UberX cars, since IDs are randomized each time a car comes online.

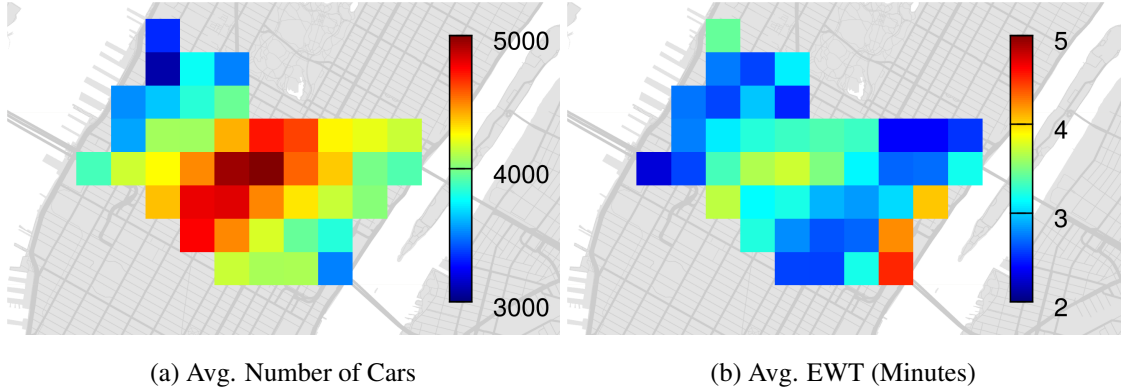


Figure 3.9: Heatmaps for UberX cars in Manhattan.

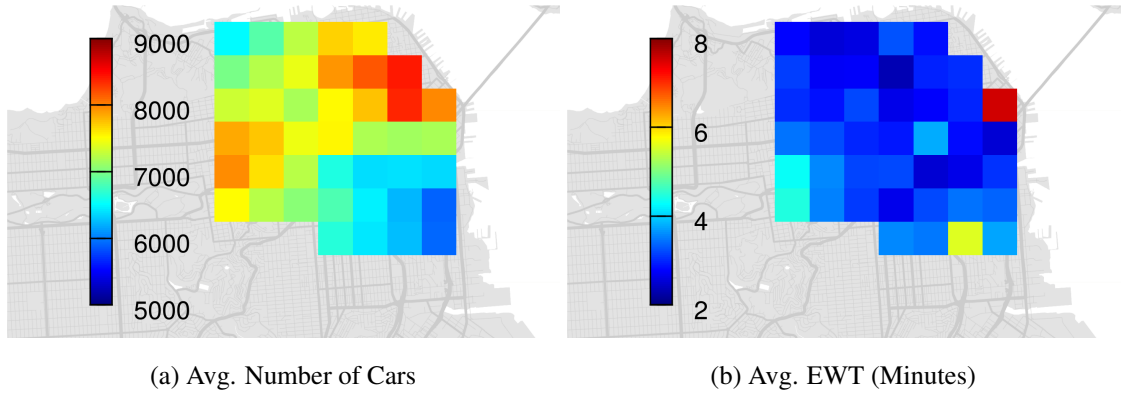


Figure 3.10: Heatmaps for UberX cars in SF.

car density and EWT. In some cases, locations with low car density have commensurately higher EWTs (e.g., the lower-right corners in Figures 3.9(b) and 3.10(b)), suggesting that these areas are under-supplied. However, I also observe under-supply in several areas with high car density (e.g., Times Square and UCSF). This complex interplay between supply and demand supports Uber’s case for implementing dynamic pricing.

Figure 3.11 presents the overall distribution of EWTs in Manhattan and SF for UberXs. 87% of the time, the wait for an UberX is ≤ 4 minutes. However, there are rare instances of severe supply/demand imbalance when EWT can go as high as 43 minutes. Note that my EWT data comes from the hearts of Manhattan and SF, and may not be representative for suburban areas.

CHAPTER 3. UBER’S SURGE PRICING ALGORITHM

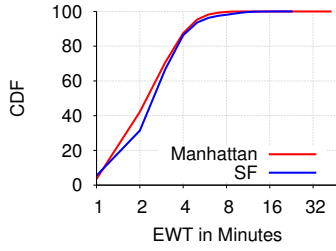


Figure 3.11: Distribution of EWTs for UberXs.

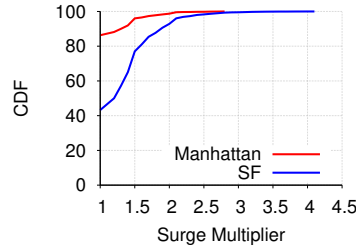


Figure 3.12: Distribution of surge multipliers for UberXs.

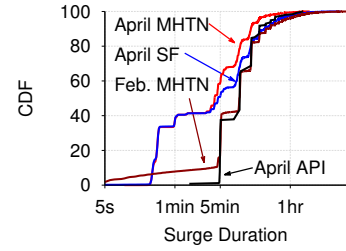


Figure 3.13: Duration of surges for UberXs.

3.4 Surge Pricing

Now that I have an understanding of supply and demand on Uber, I turn my attention to surge pricing. Surge pricing is one of Uber’s most controversial features [128, 144], and thus it warrants special attention. I begin by analyzing the basic characteristics of surge pricing. Next, I use my measured data to extrapolate how Uber updates surge prices over time and geography, and explore the features that Uber uses when calculating surge. Finally, I examine the impact of surge pricing on car supply and passenger demand.

3.4.1 The Cost of Surges

I begin by answering the questions: *how often and how much does it surge on Uber?* Figure 3.12 presents the distribution of surge multipliers for UberXs over two weeks. I observe that Manhattan and SF have drastically different characteristics: 86% of the time there is no surge in Manhattan, versus 43% of the time in SF. Furthermore, the maximum surge multiplier I observe in Manhattan is 2.8, versus 4.1 in SF. In both cities, the surge multiplier is ≤ 1.5 during the majority of surges.

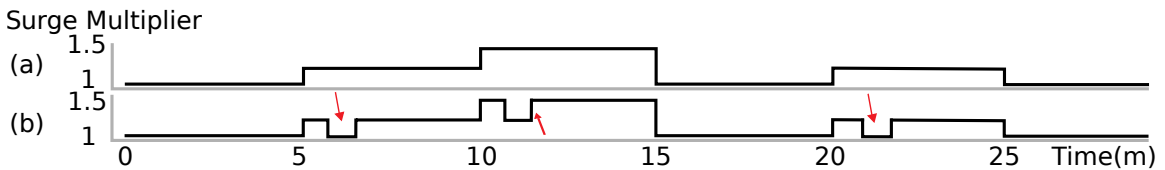


Figure 3.14: Examples of surge over time as seen from (a) the API and (b) the Client app. Although surge is recalculated every 5 minutes in both cases, clients also observe surge jitters for 20-30 seconds (red arrows).

CHAPTER 3. UBER’S SURGE PRICING ALGORITHM

The results in Figure 3.12 reveal that prices surge on Uber a large fraction of the time (in SF, it’s surging the *majority* of the time). Although most of the time the multiplier makes UberXs 25-50% more expensive, there are times (especially in SF) when the multiplier can double, triple, or even quadruple prices.

3.4.2 Surge Duration and Updates

Next, I answer the question: *how long do surges last?* Figure 3.13 plots the duration of surges for UberXs over two weeks. I define the duration of a surge as the continuous length of time when the multiplier is >1 .

When I first began collecting data from Uber in February 2015, I observed that 90% of surges had durations that were a multiple of 5 minutes. This is shown by the stair-step pattern in the “Feb. Manhattan” line in Figure 3.13. I also measured surge durations using the Uber API in April 2015, and observed the same 5-minute pattern. In both of these cases, $\sim 40\%$ of surges last 5 minutes, and $\sim 20\%$ last 10 minutes. Less than 10% of surges last more than 20 minutes.

Strangely, the behavior of the surge price algorithm changed in April 2015. As shown by the two “April” lines in Figure 3.13, 40% of surges are now less than 1 minute long. The remaining 60% of surges still loosely follow the 5-minute stair-step pattern.

There are two takeaways from Figure 3.13. First, under normal circumstances, Uber appears to update surge prices on a 5-minute clock (I discuss jitter in greater detail below). Second, the vast majority of surges are short-lived, which suggests that savvy Uber passengers should “wait-out” surges rather than pay higher prices.

Figure 3.14 illustrates how Uber updates surge multipliers. Figure 3.14(a) shows the datastream from the API (and what clients observed, prior to April 2015): surge changes at regular 5-minute intervals. Figure 3.14(b) shows the behavior of `pingClient` responses as of April 2015: surge multipliers are still updated on a 5-minute clock, but within each interval there may be brief periods of jitter. Jitter breaks up a 5-minute surge interval into three separate components, which explains why I observe many surge durations less than 1 minute long in Figure 3.13. These two processes are the same in Manhattan and SF, and for all types of Ubers.

Timing. Figure 3.15 examines the fine-grained timing of surge updates. I chose one day of data in February and April, divided time into 5-minute intervals starting at 12:00am, and calculated the exact moment in each interval when surge changed due to the clock and jitter. I observe that the old client-update and the current API-update mechanisms are extremely regular: updates occur

CHAPTER 3. UBER’S SURGE PRICING ALGORITHM

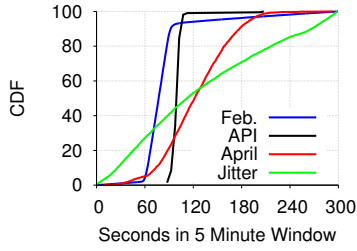


Figure 3.15: Moment when surge changes during each interval.

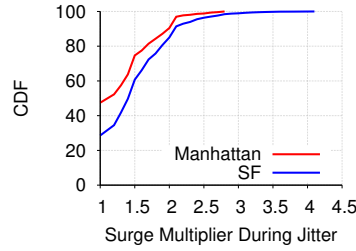


Figure 3.16: Distribution of surge multipliers seen during times of jitter.

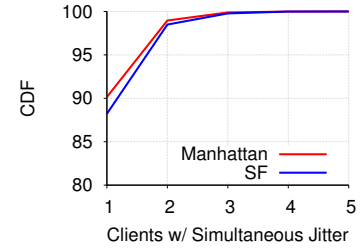


Figure 3.17: Fraction of clients that observe jitter at the same time.

during a 35-second range in each interval. The new client-update mechanism is less precise: updates occur during a 2-minute range in each interval. Jitters are distributed almost uniformly throughout the interval, which suggests that jitter is driven by a stochastic or non-deterministic process.

Jitter. Next, I examine the behavior of jitter. In my data, 90% of jitter events last 20-30 seconds, and 100% are less than 1 minute. Furthermore, during jitter, I observe that the surge multiplier is equal to the multiplier from the *previous* 5-minute interval. Because most surges only last 5 minutes, this means that jitter causes the surge multiplier to drop 74% of the time in Manhattan, and 64% of the time in SF. As shown in Figure 3.16, in 30-50% of jitter events the surge multiplier drops to 1, depending on location. Thus, jitter almost always reduces prices for passengers, if they are lucky enough to request a car during the brief window when the low multiplier is available.

Jitter also deviates from the 5-minute surge intervals in another key way. As I demonstrate in the next section, the 5-minute surge intervals are uniform over specific geographic regions. However, jitter occurs on a per-client basis. Figure 3.17 plots the number of my clients that observed jitter at the same moment in time (recall, I have 43 total clients). I see that $\sim 90\%$ of jitter events are only observed by a single client, and none are observed by more than 5 clients simultaneously.

In August 2015, I contacted Uber to make them aware of my findings. They were very concerned about the presence of jitter in the datastream, and the implication that customers were receiving inconsistent surge multipliers. I provided log data to Uber’s engineers, and they quickly determined that jitter was being caused by a consistency bug in their system. The manifestation of this bug was that random customers could receive stale surge multipliers. This precisely coincides with my observations that jitter appeared at random times for random clients, and that the surge multiplier during jitter was equal to multiplier from the previous 5-minute window. As of this writing,

CHAPTER 3. UBER’S SURGE PRICING ALGORITHM

Uber is fixing the bug and restoring consistency for all customers.

3.4.3 Surge Areas

Next, I answer the question: *how do surge prices vary by location?* Intuitively, it is clear that surge must vary geographically: a rural location is very different than Times Square.

To answer this question, I used the Uber API to query surge prices throughout Manhattan and SF over the course of eight days. During these tests, I queried data from adjacent locations that obey my visibility radius constraints (see §3.2.4). Fortunately, since I know that surge prices only change every 5 minutes (and the API does not contain jitter), this enabled me to scale up my measurements to large geographic areas.

Using this data, I looked for clusters of adjacent locations that *always* had equal surge multipliers. Figures 3.18 and 3.19 show the regions of Manhattan and SF where the surge multipliers are always in lock-step. These figures reveal the granularity of Uber’s surge price algorithm: Uber partitions cities into *surge areas* and computes multipliers independently for each area. The numbered areas correspond to the locations where I collected client data (see Figures 3.3a and 3.3b). Given the odd shape of some regions, it is likely that Uber defines surge areas manually. Note that there are some areas where I did not observe sufficient surging samples during my measurements (e.g., Upper Manhattan and Washington Heights); in these cases I cannot isolate individual surge areas, so it is possible these large areas may be composed of several smaller areas.

3.4.4 Algorithm Features and Forecasting

Next, I investigate the question: *what features does Uber use to calculate surge prices?* Uber has a pending patent that lists potential features [141], but it is unclear what features are truly used in the calculation.

To answer this question, I examine the cross correlation between observed supply, demand, and EWT versus surge price. In these tests, I treat each feature as a continuous time series. For surge, the time series is simply the observed surge multipliers during each 5-minute interval (I discard jitters since they are unpredictable). For supply, demand, and EWT I construct corresponding time series by averaging each quantity over the 5-minute window. I construct independent time series for the four areas in Manhattan and SF where I collect client data, since each area has its own surge characteristics. As before, I focus on UberXs.



Figure 3.18: Surge areas in in Manhattan.

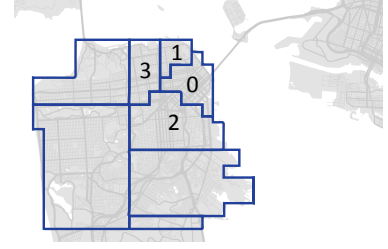


Figure 3.19: Surge areas in SF.

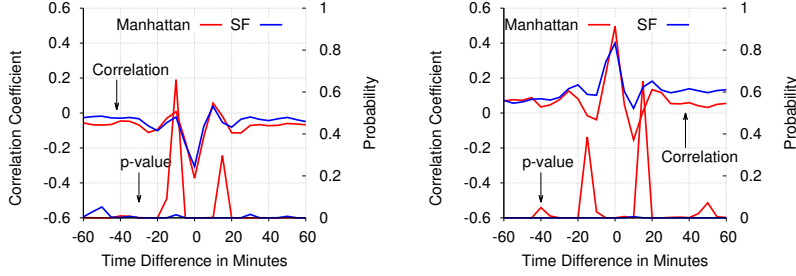
Although I examined many possible correlations between these features (such as supply and surge, demand and surge, supply-demand ratio and surge, *etc.*), I obtained the strongest results when comparing (*average supply - average demand*) and *average EWT* to surge. Figure 3.20 shows the cross correlation (and p-value) between supply/demand difference and surge price. The correlation coefficient at time shift Δt is computed using surge at time t and feature values in the interval $[t + \Delta t - 5, t + \Delta t)$. I observe a relatively strong negative correlation when $-10 \leq \Delta t \leq 10$, which indicates that the surge multiplier rises when supply/demand difference shrinks. It appears that Uber's goal is to maintain a certain amount of slack in their car supply: when demand approaches available supply, surge pricing is instituted to increase supply and reduce demand. Figure 3.20 also reveals that Uber's surge pricing algorithm is quite responsive, since the correlation is strongest when $\Delta t = 0$.

Figure 3.21 shows the cross correlation between average EWT and surge price. In this case, I see a relatively strong positive correlation at $\Delta t = 0$, i.e., EWT and surge price increase at the same time. This result also makes sense: if surge increases during times of strained supply, then the wait times for cars should also increase.

Forecasting. Given that I observe correlations between supply, demand, EWT, and surge prices, this raises a new question: *can future surge prices be forecast?* To answer this question, I fitted three linear regression models that take the current supply/demand difference, EWT, and surge multiplier for UberXs as input, and predict the surge multiplier in the next 5-minute interval. As before, I filter jitter out of the time series. I also evaluated models that accept historical data (e.g., samples from previous 5-minute intervals), but this resulted in worse predictive performance.

Table 3.1 presents the overall R^2 scores for three linear regression models evaluated on two weeks of UberX data. I fitted separate models for all four surge areas in Manhattan and SF (see Figures 3.18–3.19), but in the interest of space I present the average R^2 scores in Table 3.1. In

CHAPTER 3. UBER’S SURGE PRICING ALGORITHM



City	R^2 Score		
	Raw	Threshold	Rush
NY	0.37	0.43	0.43
SF	0.40	0.43	0.57

Table 3.1: R^2 scores of linear regression in Manhattan and SF

Figure 3.20: (Supply - Demand) vs. Figure 3.21: EWT vs. Surge for UberX.

all three models, I remove time intervals when the surge multiplier equals 1 from the data⁷ before fitting, since this would make the prediction task too easy (e.g., you could achieve 86% accuracy in Manhattan by always predicting that the surge multiplier will be 1, see §3.4.1).

Unfortunately, even with my large corpus of data, forecasting surge multipliers is an extremely difficult task. The *Raw* model in Table 3.1 is the most permissive of my three models: it was fitted and evaluated on the entire time series. I see that it has the worst performance, which suggests that either I am missing some key data that is used in Uber’s surge calculation, or surge pricing is simply very noisy. The *Threshold* model improves performance by being more restrictive: it only attempts to predict the surge at time t if surge was >1 at $t - 1$. I instituted this filter because surge cannot go below 1, i.e., I know less about the state of the system when surge is 1 than when surge is >1 .

Finally, the *Rush* model is the most restrictive: it was fitted and evaluated on the subset of data corresponding to rush hours (6am–10am and 4pm–8pm). Although the performance of the Rush model clearly benefits from the predictable characteristics of rush hour traffic (see §3.3.2), it does not perform uniformly better than the more general Threshold model.

In summary, my forecasting results demonstrate that it is very difficult to predict surge multipliers, even with large amounts of supply and demand data. None of my models exhibits strong predictive performance in absolute terms (i.e., $R^2 \geq 0.9$). This suggests that Uber relies on non-public data to calculate surge prices, and motivates me to examine alternative strategies for obtaining lower prices in §3.5.

⁷The two exceptions to this data cleaning rule are intervals where surge=1 directly preceding or proceeding an interval where surge >1 .

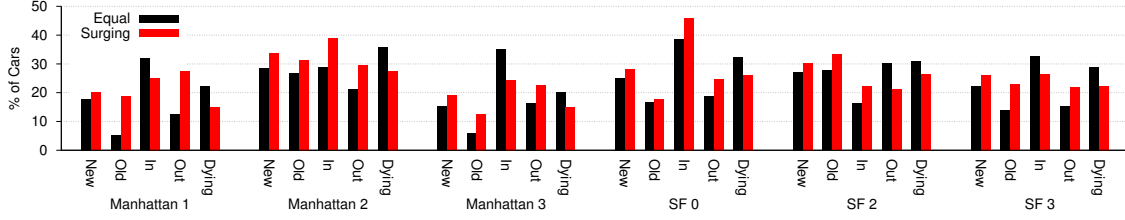


Figure 3.22: Transition probabilities of UberXs when all areas have equal surge, and when one area is surging.

3.4.5 Impact of Surge on Supply and Demand

The final question that I address in this section is: *what is the impact of surge prices on supply and demand?* Uber has stated that the goals of surge pricing are to increase supply and intentionally reduce demand, and they claim that the system increased the number of drivers by 70-80% after it was introduced [84]. However, it is unclear if and how surge pricing continues to impact supply and demand, now that drivers and passengers have acclimated to the system. Furthermore, recent measurements of Uber suggest that surge pricing redistributes existing supply, rather than encouraging new drivers to come online [67], but these observations have not been verified at-scale.

To answer these questions, I treat the cars in my data as state-machines, and examine how they transition between states when there is and is not surge. At a high-level, I divide time into 5-minute intervals, and compare the states of cars at the beginning and end of each interval. Cars that appear for the first time in interval t are placed in the initial, **new** state, while cars that disappear go into the terminal **dying** state. Cars that start and end in surge area a are **old**. Finally, cars may transition into states that are relative to surge areas, e.g., a car that moves from area a_i to area a_j during t is placed in the move-**in** state relative to a_j , and the move-**out** state relative to a_i .

Based on this model, I examine the behavior of cars during times when all four surge areas have the *same* surge multipliers (in Manhattan and SF, respectively), and times when a single area has a surge multiplier that is at least 0.2 *higher* than its neighboring areas (again, excluding jitter) in the immediately preceding interval. Intuitively, the former case captures times when there is no monetary incentive for drivers to choose one area over another, while in the latter case there is a monetary incentive to relocate to the surging area.

Figure 3.22 shows the probability of cars in specific surge areas being in each of my five states. The black bars show the probabilities when all areas have equal surge multipliers, while red corresponds to times when the given area has a multiplier that is at least 0.2 higher than its neighbors.

CHAPTER 3. UBER'S SURGE PRICING ALGORITHM

For example, the Manhattan Area 1 “New” bars show that 18% of new cars appear in the area when surge is equal in all four Manhattan areas, whereas 20% of cars appear in the area when it has higher surge than its neighbors. I omit results for two areas because they rarely had higher surge prices than their neighbors.

Results. First, I examine the impact of surge pricing on the behavior of **new** cars. In all six areas, I observe that the fraction of **new** cars increases when that area has higher surge than its neighbors. Although this effect is not large (3.7% on average), it is consistent. This suggests that surge pricing is effective at drawing drivers onto the roads.

Second, I examine the impact of surge on the distribution of existing supply. In five areas, I observe that the fraction of cars that move **out** of an area increases when it is surging. This is the opposite result of what I expected. Furthermore, in three areas (Manhattan 2, SF 0, and SF 2) I observe more cars moving **in** during times of surge, while in three other areas (Manhattan 1, Manhattan 3, and SF 3) I observe less cars moving **in** during surges. This result is inconclusive, and also unexpected: I assumed that cars would flock to the surging area.

Third and finally, I examine the impact of surge on passenger demand. In all six areas, the fraction of **old** cars increases while **dying** cars decrease within the surging area. Both of these observations point to reduced demand within the surging area.

Discussion. The results in Figure 3.22 paint a complex picture of surge pricing's impact on supply and demand. On one hand, surge does seem to have a small effect on attracting new cars. On the other hand, it also appears to have a larger, negative effect on demand, which causes cars to either become idle or leave the surge area. Although I cannot say with certainty why surge has such a large, negative effect on demand, one possibility is that customers have learned that surges tend to have short duration (see Figure 3.13), and thus they choose to wait for 5 minutes before requesting a ride. Another possibility is that surging areas may be impacted by adverse traffic conditions, which prevents drivers from flocking to them.

To make the surge pricing algorithm more effective, I propose that Uber alter the algorithm to update surge prices more smoothly. For example, rather than oscillating between periods of no and high-surge, Uber could use a weighted moving average to smooth the price changes over time. This would make surge price changes more predictable and less dramatic, which may encourage driver flocking, as well as discourage sudden, temporary drops in customer demand. Another alternative would be for Uber to adopt Sidecar's pricing approach, in which drivers set their own prices independently. This free-market approach obviates the need for a complex, opaque algorithm

and empowers customers to accept or decline fares at will.

3.5 Avoiding Surge Pricing

In the previous section, I show that short-term surge prices cannot be forecast, even with large amounts of data directly from Uber. This is disappointing, since forecasting short-term changes in surge prices would be a useful capability for drivers and passengers.

In this section, I propose an alternative method that passengers can use to obtain lower prices from Uber. Since I cannot forecast surges, this means that only the price information during the current 5-minute surge interval is reliable. Thus, my goal is to locate the lowest price car for the passenger given their current location and the instantaneous surge prices.

My Approach. My key insight is to leverage my knowledge of surge areas. Suppose a user observes that the surge multiplier at their current location is m' , and there are a set of adjacent surge areas A . I can use the Uber API to query the surge multiplier m_a and EWT e_a for each $a \in A$, as well as the walking time w_a to each area. If $m_a < m'$ and $w_a \leq e_a$ for some a , then this means the user could reserve an Uber immediately at a lower price, and walk to the pickup point in the adjacent area before the car arrives. Startups have proposed similar approaches to avoiding surge prices [163], however their techniques do not leverage precise knowledge of surge areas, or take EWTs into account.

Results. To demonstrate the feasibility of my approach, I plot Figure 3.23, which shows the percentage of time that each of my 43 measurement clients could have obtained a cheaper UberX using my approach. I assume that people can walk 83 meters per minute (i.e., 5km/hour). I also assume that, for my technique to be implemented in practice, it would need to rely on data from the Uber API. Thus, surge prices change every 5 minutes and there is no jitter, but EWTs may change moment to moment.

In Manhattan, I see that users around Times Square would have been able to request cheaper cars 10-20% of the time, depending on the user’s precise location. In contrast, users in SF would not generally benefit from my approach; only users at UCSF would be able to save money 2% of the time. My approach works better in Manhattan for two reasons: first, the surge areas in Manhattan are smaller, so it is more feasible for users to walk from one area to another in only a few minutes (50% of EWTs on Uber are less than 3 minutes, see Figure 3.11). Second, the surge areas in SF tend to be more correlated than those in Manhattan, i.e., it’s rare for one area in downtown SF to

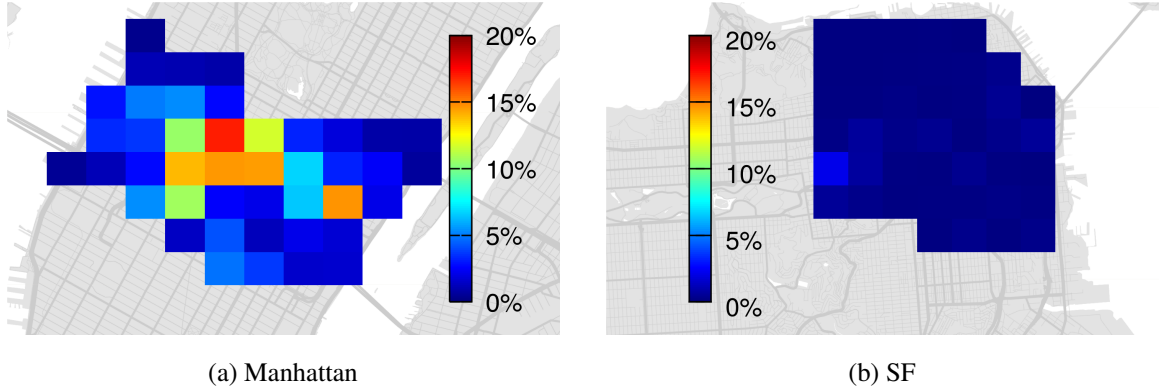


Figure 3.23: Fraction of the time when passengers would receive lower surge prices on UberXs by walking to an adjacent area.

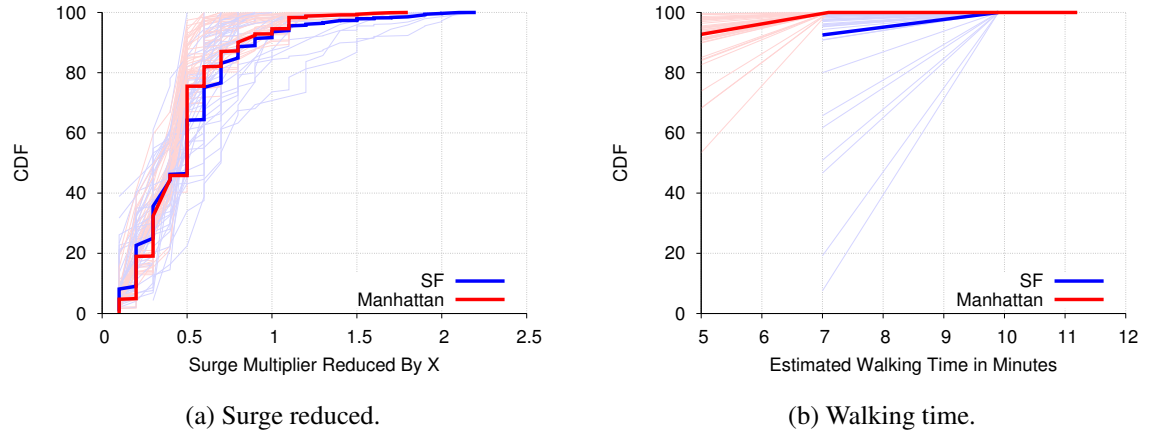


Figure 3.24: Amount that surge is reduced and walking time needed when passengers walk to an adjacent area.

have significantly higher surge than all the others.

Figure 3.24(a) shows how much surge multipliers are reduced when users reserve Ubers using my approach. The faded lines correspond to my 43 clients in Manhattan and SF, while the solid lines are the combined results. I see that my approach brings substantial savings: in more than 50% of cases, surge multipliers are reduced by at least 0.5, i.e., a 50% savings or more.

Figure 3.24(b) shows how many minutes users would need to walk while using my approach. In all cases, users walk for less than 7 and 9 minutes to meet the car in the adjacent surge area in Manhattan and SF, respectively. Note that the surge areas in SF are larger than those in Manhattan, so the shortest walks in SF are 7 minutes long, versus 5 minutes in Manhattan. Overall,

CHAPTER 3. UBER’S SURGE PRICING ALGORITHM

these walking times are quite reasonable, given the dramatic savings that this strategy enables.

3.6 Summary

In this chapter, I present an in-depth analysis of Uber. I leverage four weeks of data collected from Uber’s smartphone app and official API that covers midtown Manhattan and downtown SF. In total, I collected 2.1 TB of data on supply, demand, EWTs, and surge prices for Uber vehicles. I validate the fidelity of my methodology using ground-truth information about all taxi rides in NYC in 2013.

My results reveal high-level characteristics about Uber’s service and the impact of surge pricing. I observe that SF has $3\times$ more surges than Manhattan even though SF also has a much greater supply of cars. Furthermore, surge pricing is noisy: the majority of surges last less than 10 minutes. Alarming, my investigation uncovered a bug in Uber’s systems that was causing some customers to randomly receive out-of-data surge price information. Finally, I observe that on a micro-scale, surge prices have a strong, negative impact on passenger demand, and a weak, positive impact on car supply. However, it is possible that different effects may occur at the macro-scale (i.e., a whole city).

Chapter 4

Algorithmic Pricing on Amazon Marketplace

In this study, my goal is to empirically analyze deployed algorithmic pricing strategies on Amazon Marketplace. Specifically, I want to understand what algorithmic pricing strategies are used by participants in the market, how prevalent these strategies are, and ultimately how they impact customer experience.

The remainder of this study is organized as follows. §4.1 covers background on Amazon and the Amazon Marketplace, and technical features that facilitate algorithmic pricing. In §4.2, I present my data collection methodology, including specific challenges that I needed to overcome to obtain useful, representative data. §4.3 explores the algorithm that Amazon uses to select the Buy Box winner, and shows the features that drive the Buy Box algorithm. §4.4 presents my approach for detecting sellers using algorithmic pricing, and §4.5 explores the characteristics and impact of algorithmic and Prime sellers. In §4.6, I present my purchase, revenue, and Buy Box simulation methodologies and discusses their results. Finally, §4.7 summarizes the key findings of this study.

4.1 Background

I begin by briefly introducing Amazon Marketplace. I focus on the features of the market that are salient to algorithmic pricing, including Third-Party (3P) sellers, the Buy Box, and finally the APIs offered by Amazon Marketplace Web Services.

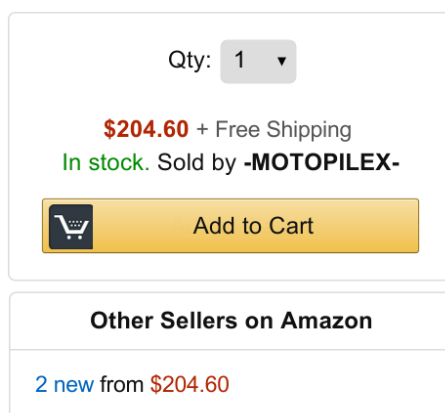


Figure 4.1: An example Buy Box on Amazon.

4.1.1 Amazon Marketplace

Amazon, founded in 1994, is the largest e-commerce website in the US and Europe [120]. Although Amazon began as an online bookstore, it now sells over 20 categories of physical products (even fresh food in select cities [34]), as well as a wide range of digital goods (e.g., downloadable and streaming music, video, and e-books). Overall, Amazon earned \$89B in revenue in 2014, and boasts 244M active customers [68].

Amazon inspires fierce loyalty among customers through their Prime membership program, which gives customers free 2-day shipping (or better) as well as unlimited access to digital streams for \$99/year. It is estimated that as of July 2016, half of all Amazon customers are Prime subscribers [166]. Amazon's success is further bolstered by their branded digital devices (Kindle e-readers, tablets, phones *etc.*), which push customers towards Amazon's shopping apps. Because of these customer retention efforts, more than 50% of online shoppers navigate directly to Amazon to make purchases, rather than using search engines or visiting competing online retailers [170].

3P Sellers and FBA. In addition to acting as a merchant, Amazon also functions as a marketplace for third parties. Amazon claims to have 2M Third-Party (3P) sellers worldwide who sold 2B items in 2014, representing 40% of all items sold via the website [7]. 3P sellers can opt to handle logistics (inventory, shipping, returns, *etc.*) themselves, or they can join the Fulfilled By Amazon (FBA) program, in which case Amazon handles all logistics.


The fee structure for 3P sellers is complicated, and involves five components [8, 10]:

1. **Seller Fee:** "Individual" sellers must pay \$0.99/item sold, or sellers may become "Pro Merchants" for \$39.99/month.

CHAPTER 4. ALGORITHMIC PRICING ON AMAZON MARKETPLACE

2. **Referral Fee:** Amazon assesses a referral fee on each product sold. The fees vary between 6-45% of the total sale price, depending on the product category. The vast majority of categories have a 15% referral fee. Amazon also enforces minimum referral fees of \$1-\$2/item.
3. **Closing Fee:** Amazon's closing fees vary based on product category, shipping method, and product weight. Media products (books, DVDs, *etc.*) have a flat fee of \$1.35/product. Other products have a \$0.45 + \$0.05/lb fee for standard shipping, or \$0.65 + \$0.10/lb for expedited shipping.
4. **Listing Fee:** High-volume sellers that list more than 2M Stock Keeping Units (SKUs, a seller-specified representation of an item) per month must pay \$0.0005 per active SKU.
5. **FBA Fee:** Sellers that use FBA must pay a \$1.04-\$10.34 packing fee per product depending on its size and type, plus variable per pound shipping fees ranging from \$0.39 for small media items, to \$124.58 for extremely heavy, irregularly shaped items.

As I discuss in § 4.4, these fees influence the dynamic pricing strategies used by 3P sellers.

FBA Prime and Seller Fulfilled Prime. As more customers have subscribed to Amazon's Prime loyalty program, the importance of being a *Prime seller* has increased. Prime sellers are highlighted visually with a small "badge" () on Amazon's website, which signifies to Prime members that purchases made from these sellers will be delivered in two days or less. Thus, Prime customers are incentivized to purchase products sold by Prime sellers. Amazon itself is a Prime seller.

There are two ways for a seller to become Prime eligible. The first and most straightforward way is for the seller to join the FBA program. In this case, Amazon handles all order logistics for the seller, and can thus guarantee the level of service that Prime customers expect. However, not all sellers are willing or able to join the FBA program. Thus, the second way is for the seller to join the Seller Fulfilled Prime program. Sellers who choose to participate in this program must have a proven track record of fulfilling orders quickly and maintaining available product inventory, among other requirements. Sellers who are accepted into the Seller Fulfilled Prime program are able to avoid paying the fees associated with FBA, leading to lower prices or increased profit margins, while enjoying the same benefits as FBA sellers.

CHAPTER 4. ALGORITHMIC PRICING ON AMAZON MARKETPLACE

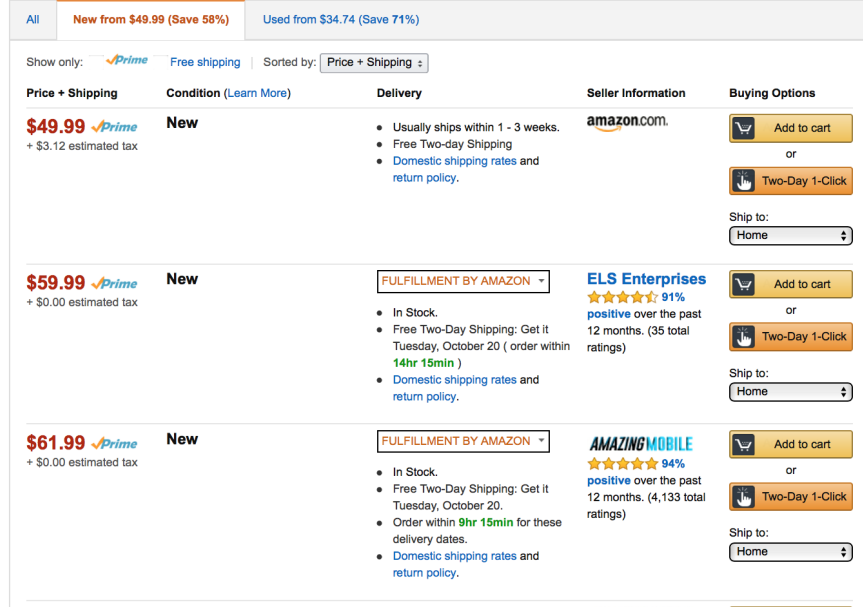


Figure 4.2: An example *New Offers* page on Amazon, listing all sellers for a given product.

4.1.2 The Buy Box

When customers purchase products from Amazon, they typically do so through the *Buy Box*. The Buy Box is shown on every product page on Amazon: it contains the price of the product, shipping information, the name of the seller, and a button to purchase the product. Figure 4.1 shows an example Buy Box.

However, many products on Amazon are sold by multiple sellers. In these cases, a proprietary Amazon algorithm determines which seller's offer is displayed in the Buy Box. Formally, if a product is being offered by n sellers with prices $P = \{p_1, \dots, p_n\}$, the Buy Box algorithm is a function $B(P) \rightarrow p_i$, with $p_i \in P$. As shown in Figure 4.1, offers from other sellers are relegated to a separate webpage (an example is shown in Figure 4.2).

Given the prominent placement of the Buy Box, it is not surprising that 82% of sales on Amazon go through it [177]. This has made the underlying algorithm the focus of much speculation by 3P sellers [31]. Although Amazon has released some information about the features used by the Buy Box algorithm (e.g., prices, shipping options, and speed) [11], it is unknown whether this feature list is complete, or what the weights of the features are.

Because “winning” the Buy Box is so critical for making sales on Amazon, sellers may use dynamic pricing strategies that give them an advantage with respect to being chosen by the algorithm.

Thus, I examine the Buy Box algorithm in-depth in § 4.3.

4.1.3 Amazon Marketplace Web Service

Amazon offers an array of tools to help 3P sellers manage product inventory. The most sophisticated of these tools is the Amazon Marketplace Web Service (MWS), which is a set of APIs for programatically interfacing with the marketplace. MWS includes functions for listing products, managing inventory, and changing prices.¹ MWS also has a subscription API, that allows sellers to receive near real-time price updates for specified products. Each update includes aggregated information about the lowest 20 prices offered for a product (or less, if there are fewer than 20 offers).

In addition to MWS, Amazon also has a web-based price matching tool for 3P sellers [13]. This tool allows a 3P seller to set a product's price equal to the lowest competing offer. However, this tool only adjusts the product's price *once*: if the lowest price changes again, the seller's price is not automatically reduced as well.

Seller Platforms. The capabilities of MWS are clearly designed to facilitate dynamic pricing. Companies like Sellery, Feedvisor, Appeagle, RepriceIt, and RepricerExpress leverage MWS to offer subscription-based services for 3P sellers that combine inventory management with dynamic pricing capabilities. These services enable any merchant to easily become a 3P seller and leverage sophisticated dynamic pricing strategies. I discuss the types of strategies offered by these services in greater detail in § 4.4.

4.2 Data Collection

There are two goals of my study: first, to analyze the dynamic pricing strategies being used by sellers on Amazon and second, to investigate the effects of pricing dynamics on buyers and sellers. To achieve this goal, I require longitudinal data about sellers and their prices—ideally for a large number of products—in the marketplace. In this section, I describe my data collection process, including specific challenges that I needed to overcome to obtain useful, representative data.

4.2.1 Obtaining Sellers and Prices

Ideally, I would have liked to use the Amazon Marketplace Web Services (MWS) API to collect the seller and price information for products. Indeed, the API offers functionality that returns

¹Amazon's documentation stipulates that sellers may only update prices every 20 minutes [6].

CHAPTER 4. ALGORITHMIC PRICING ON AMAZON MARKETPLACE

information about the top twenty offers for a given product, including the unique IDs of the sellers, their prices, *etc.* Unfortunately, the only way to access this API for a given product is to actually list it for sale. In other words, I would have had to list thousands of products for sale on Amazon to collect large-scale data from the MWS API, which is not feasible in practice (considering that I have no actual product inventory, and no ability or desire to fulfill product orders).

Instead of using the API, I used *web scraping* to obtain information on the active sellers and their prices. Specifically, for each product I examine, I crawled the *New Offers* page² (the page that is linked to in Figure 4.1 if one clicks on “2 new”, shown in Figure 4.2). This page lists all 3P sellers, their prices, their shipping costs, and their reviews (number of reviews and average score). Unfortunately, this information is paginated into ten 3P sellers per page; I describe below how I handle cases where there are more than ten 3P sellers.

In addition to scraping the *New Offers* pages for products, I also scraped the product pages themselves. I use the data from the product pages to analyze the Buy Box algorithm in § 4.3.

Calculating Prices. It is important to note that the *New Offers* page lists both the *base price* and the *shipping price* for each seller. Throughout the thesis, when I refer to “price”, I am referring to the total price, i.e. the *base price* + *shipping price*. I do this as Amazon uses total price when users explicitly sort products by price; users cannot search or sort by base price alone.

4.2.2 Crawling Methodology

Amazon operates a complex, global, dynamic computing infrastructure to support its storefront. As such, there are several technical and methodological issues that I must address to insure that the data I collect for this study is representative and unbiased.

Temporality. Because 3P sellers (and Amazon) can change their price at any time, I need to decide how frequently I will crawl each page. To do so, I create a high-resolution dataset that will help to illuminate the tradeoff between crawling resolution and frequency. Specifically, I randomly selected 5 products from the best-seller products³ and crawled their product page and the first 2 seller pages (covering up to 20 sellers) once per minute for 3 days.

²In this study, I only focus sellers who offer new items; used items are not covered, and I leave them to future work.

³<http://www.amazon.com/Best-Sellers/zgbs>, Best-seller products come from 23 departments from Amazon, such as Appliances, Beauty, Electronics, etc. Altogether there are 1,790 best-seller products (I exclude digital goods such as e-books, downloadable music, and gift cards).

CHAPTER 4. ALGORITHMIC PRICING ON AMAZON MARKETPLACE

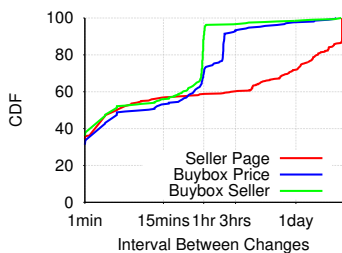


Figure 4.3: Frequency of page updates.

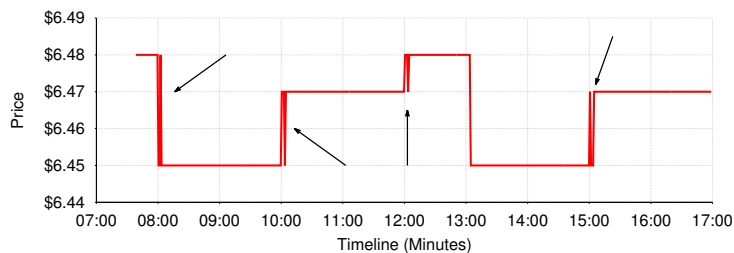


Figure 4.4: Examples of price jitter (highlighted with arrows) in the Buy Box on a product page.

I first examine how frequently sellers' prices change and how frequently the Buy Box is updated. I plot the cumulative distribution of inter-update times for sellers, the Buy Box price, and the Buy Box seller in Figure 4.3. I observe that the updates are surprisingly dynamic: 40% of price changes occur within a minute of the previous price change, with a long tail of update times. To explore the origins of this high level of dynamicity, I plot a timeseries of the Buy Box price of an example product in Figure 4.4 (I observed similar behavior for other products, but do not include them due to space constraints).

I observe that the price appears to change five times in this timeseries, but that old prices sometimes briefly reappear after a price change. This result is likely due to Amazon's distributed infrastructure; Amazon states that it can take up to 15 minutes for all systems to converge to the new price. Thus, the very rapid price "jitters" are likely caused by transient inconsistencies in Amazon's infrastructure, rather than actual price changes by sellers.⁴

Using these results, I select a crawling frequency. As a tradeoff between number of products and crawling frequency, I choose to cover more products at longer intervals. As shown in Figure 4.3, most changes happen either on very short timescales (< 1 minute; likely Amazon inconsistencies) or very long timescales (> 30 minutes). I therefore choose a crawling frequency of every 25 minutes.

Geolocation. Prior work has shown that e-commerce companies sometimes personalize prices for users based on their geolocation [129, 130]. If Amazon personalized prices based on geolocation it would complicate my data collection efforts, since I would need to collect data from

⁴To further verify these results, I set up an Amazon Individual Seller account, listed several products, and changed their prices at specific times. I found that when prices are in an inconsistent state, a customer cannot add the item to their shopping cart (i.e., even though a customer may see an outdated price, the customer is not able to add the product to their cart at the old price).

CHAPTER 4. ALGORITHMIC PRICING ON AMAZON MARKETPLACE

diverse geolocations to obtain a representative sample of prices.

To determine if Amazon personalizes prices based on geolocation, I crawled 20 products every hour for three days from three different locations around the US. Then I compared the Buy Box winner and Buy Box price of each product in each time epoch across the three locations. *I did not observe any differences in the Buy Boxes across my three geolocations.*

Based on this result, I conclude that prices on Amazon are consistent regardless of the users' location (at least with respect to users visiting `amazon.com` from within the continental US). Thus, I do not consider geolocation further in this study.

Prime Membership. Another important question is whether Amazon displays different Buy Boxes to Prime and non-Prime users. In theory, Amazon could personalize the Buy Boxes for Prime users to always show them an offer from a Prime seller. Intuitively, this might encourage Prime customers to make more purchases.

To determine if Amazon personalizes the Buy Box based on the Prime-status of the user, I conducted two crawls in parallel: one from a browser that was not logged-in to Amazon, and a second browser that was logged-in to a Prime account. Each browser crawled the product pages and first two seller pages for 1,000 products, revisiting each product every 25 minutes. I repeated this process for 15 days. Finally, I compared the Buy Box seller, Buy Box price, and top-20 seller list of each product in each time epoch across the two browsers. *I did not observe any differences in the Buy Boxes or seller lists between my Prime and non-Prime user accounts.*

Based on this result, I conclude that Amazon does not personalize the Buy Box based on the Prime-status of the user. Note that Amazon may personalize other aspects of their storefront based on the user's Prime-status (e.g., product recommendations and search results), but these forms of personalization do not impact my data collection in this study.

4.2.3 Selecting Products

Next, I turn to selecting the products to study. Recall that I am aiming to study dynamic pricing; not all products are equally likely to have such sellers, so I focus on best-selling products since they are likely to have many competing sellers. I conduct two separate crawls that have different characteristics.

First Crawl (*CrawlI*). My first crawl was conducted between September 15, 2014 and December 8, 2014. I select 837 best-selling products that had at least two sellers at the beginning

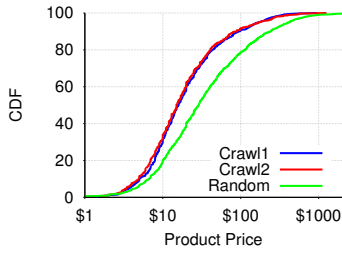


Figure 4.5: Cumulative distribution of product prices.

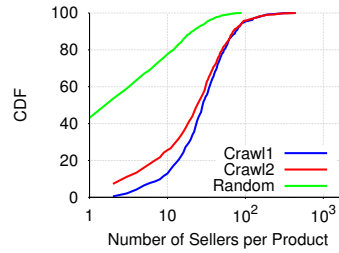


Figure 4.6: Cumulative distribution of number of sellers per product.

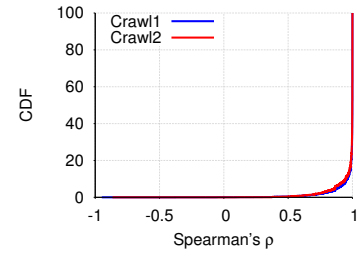


Figure 4.7: Correlation between price and rank (-1 is anti-correlation, 1 is perfect correlation).

of the crawl. For this crawl, I downloaded *all* seller pages, but did not download the product page (containing the Buy Box).

Second Crawl (Crawl2). I conduct a second crawl between August 11, 2015 and September 21, 2015. I select 1,000 best-selling products to study, and downloaded both the product page (containing the Buy Box) and the first two pages of 3P sellers (typically, but not always, containing the 20 sellers with the lowest prices). I choose to only download the first two pages of sellers, as I found the sellers who change their prices often (suggesting dynamic pricing algorithms) were within the first two pages 96% of the time. Thus, downloading only the first two pages massively reduces the amount of data I need to collect while still capturing most of the “interesting” behavior.

It is important to note that these crawls cover different products, as the best-selling products change over time. As shown in Figures 4.5 and 4.6, the overall characteristics of prices and sellers are very similar between the first two crawls despite the time difference (details of these Figures are discussed in the next section).

Limitations. There is one noteworthy limitation to my dataset: it is biased (by design) towards best-selling products. To briefly quantify this bias, I randomly sampled 2,158 products from a public listing of all Amazon products.⁵ I compare the product price and the number of sellers in Figures 4.5 and 4.6; as expected, I observe that my best-sellers show many more sellers than random products, as well as somewhat lower prices.

⁵https://archive.org/details/asin_listing

4.3 The Buy Box

I begin my analysis by exploring how Amazon’s systems evaluate sellers. *First*, I briefly examine Amazon’s seller ranking algorithm, and follow up by characterizing the dynamics and behavior of the Buy Box. In both cases, I observe that Amazon uses non-trivial strategies to evaluate sellers (i.e., price is not the only factor that impacts ranking and selection for the Buy Box). *Second*, I conduct an in-depth investigation of the features and weights that drive the Buy Box algorithm. Understanding the Buy Box algorithm is crucial, since it may influence how sellers choose dynamic pricing strategies.

Note that in this section, I only use data from Crawl2, since it contains Buy Box winners and seller rankings.

4.3.1 Seller Ranking

As shown in Figure 4.2, Amazon explicitly *ranks* all sellers for each product on the *New Offers* page. However, the Buy Box winner is not necessarily the seller who is ranked the highest. Thus, I first examine the seller ranking algorithm as it offers clues as to how Amazon chooses to weigh various seller features.

I collect the rankings for all products in my dataset, and calculate Spearman’s Rank Correlation (ρ) between the ordered list of sellers returned by Amazon, and the list of sellers sorted by price, for each product in my dataset. If the lists perfectly correspond (i.e., Amazon returns sellers sorted by price), then Spearman’s ρ will equal 1. Contrary to my expectations, Amazon does not always sort sellers by price. As shown in Figure 4.7, around 20% of products have correlation < 1 . This result gives me my first clue that Amazon’s systems take additional seller attributes into account (besides price) when making decisions.

4.3.2 Behavior of the Buy Box Algorithm

Next, I examine the empirical behavior of the Buy Box, starting with dynamics over time. Figure 4.8 plots the number of changes to the price and seller in the Buy Box for all products in the Crawl2 dataset over the six weeks of observation. I immediately see that most products change a number of times: only 13% of products have static Buy Box prices over the entire period, while 50% of products have more than 14 changes. However, I see fewer changes to the Buy Box winner: the seller winning the Buy Box is constant for 31% of products. Thus, for many products, the Buy Box

CHAPTER 4. ALGORITHMIC PRICING ON AMAZON MARKETPLACE

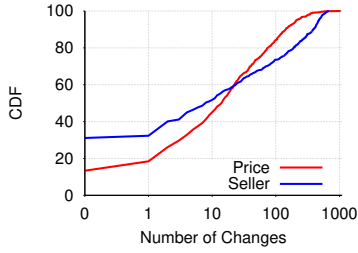


Figure 4.8: Cumulative distribution of changes to the Buy Box price and winner, per product.

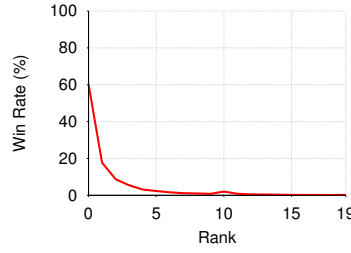


Figure 4.9: Probability of winning the Buy Box for sellers at different ranks.

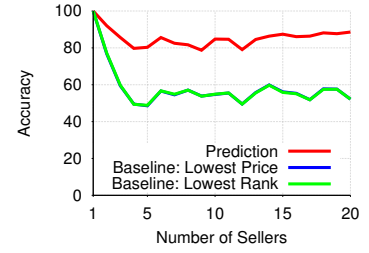


Figure 4.10: Buy Box winner prediction accuracy for products with different numbers of sellers.

winner and price is highly dynamic; some products even experience hundreds of changes, or many more than one per day.

Next, I examine the relationship between seller rank (from the *New Offers* page) and the winner of the Buy Box. Figure 4.9 shows the fraction of sellers at different ranks that “win” the Buy Box, i.e., are chosen by the algorithm. Rank zero means they are the first seller in the list. Surprisingly, only 60% of the top-ranked sellers win the Buy Box, and there is a long tail of sellers at higher ranks that win. Recall that I have already shown that Amazon does not rank sellers solely on prices (see Figure 4.7). Taken together, these results show that Amazon’s systems take additional characteristics beyond price into account when evaluating sellers.

4.3.3 Algorithm Features and Weights

In the previous section, I demonstrated that the Buy Box algorithm uses features beyond just price to select the Buy Box winner. In this section, I use Machine Learning (ML) and regression analysis to try to infer some of the features and weights used by the Buy Box algorithm.

Model and Features. To facilitate my analysis, I model the Buy Box as a prediction problem. Specifically, for a product offered by n sellers, each of which is characterized by a feature vector, my goal is to predict which seller will be chosen to occupy the Buy Box. Given my dataset, I construct a feature vector for each seller containing the following seven features:

1. *Price Difference to the Highest*: difference between the seller’s price and the current highest price for the product.

CHAPTER 4. ALGORITHMIC PRICING ON AMAZON MARKETPLACE

2. *Price Ratio to the Highest*: ratio between the seller's price and the current highest price of the product.
3. *Average Rating*: average customer rating⁶ of the seller.
4. *Positive Feedback*: positive feedback percentage for the seller.
5. *Feedback Count*: total feedback count for the seller.
6. *Is the Product FBA?*: true if the seller uses FBA.
7. *Is Amazon the Seller?*: true if the seller is Amazon.

According to Amazon's documentation, as well as speculation from 3P sellers, other features are possibly used by the Buy Box algorithm [11,31]. This includes sales volume, response time to customer inquiries, rate of returns and refunds, and shipping times. Unfortunately, I cannot measure these features, and thus cannot quantify their impact on the Buy Box algorithm. However, as I will show, even without these features I am able to achieve high prediction accuracy, suggesting that my data does capture the most important seller features.

Classifier Selection. I leverage a Decision Tree (DT) classifier to predict whether a specific seller of a product wins the Buy Box. DT is a robust, non-linear classifier with the ability to handle both categorical and continuous variables while exploiting various feature combinations [172]. Furthermore, DT is an ideal classifier for my task because it outputs interpretable measures of feature importance (calculated as the Gini index). In contrast, other classifiers, such as kernel-based SVM, are harder to interpret. Note that I evaluated other common classifiers, such as Random Forest and SVM, but found that DT performed best in my experiments.

Evaluating the Classifier. Figure 4.10 shows the accuracy of my DT classifier at predicting the winner of the Buy Box (using 10-fold cross-validation) for all products in Crawl2, as a function of the number of sellers for a given product. Obviously, it is trivial to achieve 100% accuracy in the 1-seller case; however, I see that the classifier achieves 80–87% accuracy even in the most challenging cases with many sellers.

To put the accuracy results of my classifier in perspective, Figure 4.10 also depicts the accuracy of two naïve *baseline* classifiers. One baseline classifier always predicts that the seller with

⁶Fully described in § 4.5.1, the rating and feedback of a seller come from customer surveys asking for a star rating (0–5 stars) and specific questions about the customer's experience.

CHAPTER 4. ALGORITHMIC PRICING ON AMAZON MARKETPLACE

Feature	Decision	Logistic Regression	
	Tree Weight	Coeff	Std-Err
Price Ratio to the Highest	0.34	-0.43***	0.003
Price Difference to the Highest	0.30	0.01	0.009
Positive Feedback	0.12	0.44***	0.008
Feedback Count	0.10	-0.002	0.007
Average Rating	0.06	0.26***	0.003
Is Amazon the Seller?	0.06	—	—
Is the Product FBA?	0.03	0.44***	0.003

Table 4.1: Relative importance of different features in winning the Buy Box, as determined by my DT classifier. The significance of coefficients in the regression model are: * $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$.

the lowest price will win the Buy Box (if there are multiple sellers offering the same lowest price, it chooses the lowest ranked one), while the other chooses the lowest ranked seller. Both baselines only achieve 50–60% accuracy, which reconfirms that price is not the sole feature used by the Buy Box algorithm, and also highlights the impressive predictive power of my DT classifier.⁷

Feature Importance. Next, I examine the Gini importance calculated for each feature by my DT classifier. Higher feature-importance implies that the feature is more predictive in deciding who wins the Buy Box. As shown in Table 4.1, the two price-based features are significantly more important than other features. However, the seller’s positive feedback and feedback count are also important metrics for winning the Buy Box.

Although I observe that “being Amazon” does confer some advantage, I caution that this does not necessarily mean that Amazon has tilted the Buy Box algorithm in their favor. Recall that Amazon’s Buy Box documentation states that the algorithm uses several features that I am unable to measure, such as sales volume [11]. It is possible that Amazon scores highly in these missing features, which manifests in my classifier as additional weight in the “Amazon is the Seller” feature.

Interestingly, I observe that the DT classifier assigns low weight to the FBA feature. This struck me as odd, given that conventional wisdom about how the Buy Box algorithm functions [31] suggests that FBA should have positive correlation with winning the Buy Box. To explore this discrepancy, I examine feature importance using standard regression tests in the next section.

Regression Analysis. The feature importance scores returned by the DT classifier

⁷My dataset includes at least 1K samples at each rank, thus the results are statistically significant.

CHAPTER 4. ALGORITHMIC PRICING ON AMAZON MARKETPLACE

should not be interpreted as the correlation or “weights” of the features. A feature is deemed important by the DT learning algorithm based on its ability to split the data while decreasing entropy.

To determine the weight or correlation of the features, I ran a logistic regression on the data. The dependent variable is binary (whether or not the seller won the Buy Box) and the independent variables are kept the same as the DT features with two exceptions: I exclude the “*Is Amazon the Seller?*” feature, and I add a constant unit intercept. For the regression tests, I omitted all data points where Amazon is the Buy Box seller since the standard deviation for Amazon winning the Buy Box is always close to zero (Amazon wins the Buy Box 90% of the times it contests for it). This explains I why omitted the “*Is Amazon the Seller?*” feature from the regression.

The results of the regression are shown in Table 4.1. The coefficients for features *Price Difference to the Highest* and *Feedback Count* are inadmissible due insignificant p-values. Of the remaining features, *FBA*, *Positive Feedback* and *Price Ratio to the Highest* have the highest correlations. In this case, the *FBA* feature coefficient obeys conventional wisdom (i.e., it is important), which stands in stark contrast to the feature importance returned by the DT classifier. This demonstrates that while having FBA improves a seller’s odds to win the Buy Box, it is not enough to determine a unique winner (which is not surprising, since it is only a binary variable). To give a more concrete example, consider a scenario where 15 out of the 20 sellers are offering FBA. Given the strong, positive correlation between FBA and winning the Buy Box, there is a high probability that the Buy Box winner is one of the 15 sellers. However, for selecting a Buy Box winner from the 15, one has to rely on features other than FBA. In other words, although FBA shows significant positive correlation, its discriminatory power is low, which is why it is awarded a poor feature importance score by the DT classifier.

Summary. Overall, I observe that Amazon’s algorithm for choosing the winner of the Buy Box is a combination of a number of undocumented features and weights. I am able to gain some visibility into this algorithm, with the results indicating that price, seller feedback, feedback count, and FBA are all important features. These results suggest that sellers who use algorithmic strategies to maintain low prices relative to their competitors are likely to gain a large advantage in the struggle to win the Buy Box.

4.4 Dynamic Pricing Detection

I now turn to detecting algorithmic pricing on Amazon Marketplace. I note that doing so is non-trivial, as external observers such as ourselves are only able to measure the prices offered by sellers (and not their usage of the Amazon Marketplace API, *etc*). Moreover, I lack ground truth on which sellers are using algorithm pricing. Therefore, I build a detection algorithm that tries to locate sellers that *behave* like “bots”, i.e., sellers where the prices they set and the timing of changes suggest algorithmic control.

4.4.1 Methodology

I hypothesize that sellers using algorithmic pricing are likely to base their prices at least partially on the prices of other sellers. This makes sense intuitively: for example, a seller who always wants to offer the lowest on a specific product must set their price relative to the competitor with the lowest price. Thus, I first define several *target prices* that the seller could match against. I motivate my selection of target prices by examining popular repricing software for Amazon Marketplace,⁸ and choose three target prices for each product: lowest price, Amazon’s price, and the second-lowest price.⁹

For a given seller/product pair (s, r) , I construct a time series of the prices p_i offered by the seller at time t_i :

$$S_r = \{(t_0, p_0), (t_1, p_1), \dots, (t_m, p_m)\}$$

I also construct three target price time series, corresponding to the lowest price p_i^{low} , the 2nd lowest price p_i^{2nd} , and Amazon’s price p_i^{amzn} for r at each time t_i :

$$\begin{aligned} LOW_r &= \{(t_0, p_0^{low}), (t_1, p_1^{low}), \dots, (t_m, p_m^{low})\} \\ 2ND_r &= \{(t_0, p_0^{2nd}), (t_1, p_1^{2nd}), \dots, (t_m, p_m^{2nd})\} \\ AMZN_r &= \{(t_0, p_0^{amzn}), (t_1, p_1^{amzn}), \dots, (t_m, p_m^{amzn})\} \end{aligned}$$

Note that when I construct the three target price time series, I exclude the prices offered by s . For example, if s always offers the lowest price for r , then LOW_r will actually contain the second-lowest price for r at each time t_i . This exclusion rule also prevents me from comparing Amazon’s prices

⁸<https://sellerengine.com/sellery/>

⁹In fact, Sellery provides one more option: matching to the average price. However, this strategy is neither likely to be useful for winning the Buy Box, nor being competitive among the sellers.

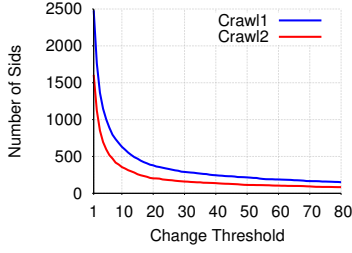


Figure 4.11: Number of algorithmic sellers detected with different change thresholds.

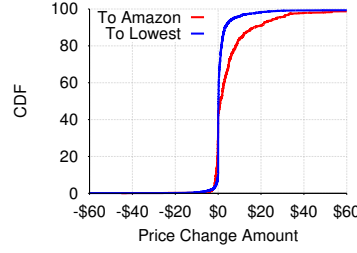


Figure 4.12: CDF of absolute price differences between algorithmic sellers and target prices.

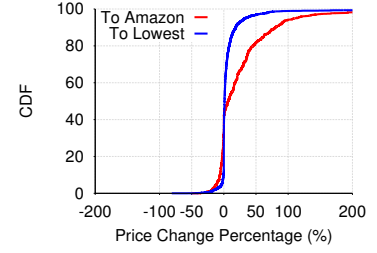


Figure 4.13: CDF of relative price differences between algorithmic sellers and target prices.

against themselves. Finally, note that Amazon does not sell all products in my dataset, thus only a subset of seller/product pairs include $AMZN_r$.

Once I have constructed the time series corresponding to (s, r) , I calculate the similarity between S_r and LOW_r , $2ND_r$, and $AMZN_r$ (respectively) using Spearman's Rank Correlation. When ρ is large, it means that the price changes contained in the pair of time series occur at the same moments, and that the magnitude of the price changes are relatively constant. I mark pairs with $\rho \geq 0.7$ (the empirical cutoff of a strong positive correlation) and $p\text{-value} \leq 0.05$ as *algorithmic pricing candidates*.

The final step in my methodology is to filter my candidates. Intuitively, if a seller exhibiting high correlation with the target price also makes a large number of price changes, this provides more evidence that the seller is using algorithmic pricing. Conversely, if the number of price changes in the time series is small, then it is possible that the correlation is coincidental. Thus, I define the *change threshold* as the minimum number of price changes that must occur in a time series S_r for me to consider s as using algorithmic pricing.

Figure 4.11 shows the number of sellers that I consider to be doing algorithmic pricing when I apply different change thresholds. As expected, I observe that the number of sellers decreases rapidly as I increase the change threshold. Unless otherwise stated, in the remainder of the study, I choose 20 as my change threshold since it represents a conservative threshold that is in the “knee” of the distribution.

The final step in my methodology is to filter my candidates. Intuitively, if a seller exhibiting high correlation with the target price also makes a large number of price changes, this provides more evidence that the seller is using algorithmic pricing. Conversely, if the number of price changes in the time series is small, then it is possible that the correlation is coincidental. Thus, I define the

CHAPTER 4. ALGORITHMIC PRICING ON AMAZON MARKETPLACE

Strategy	Threshold = 10		Threshold = 20	
	Sellers	Products	Sellers	Products
Lowest Price	726	544	426	408
Amazon Price	297	277	176	183
2nd Lowest Price	721	494	425	370
Total	918	678	543	513

Table 4.2: Number of sellers and products with detected algorithmic pricing, based on two different change thresholds. I use a change threshold of 20 unless otherwise stated.

change threshold as the minimum number of price changes that must occur in a time series S_r for me to consider s as using algorithmic pricing.

Figure 4.11 shows the number of sellers that I consider to be doing algorithmic pricing when I apply different change thresholds. As expected, I observe that the number of sellers decreases rapidly as I increase the change threshold. Unless otherwise stated, in the remainder of the thesis, I choose 20 as my change threshold since it represents a conservative threshold that is in the “knee” of the distribution.

4.4.2 Algorithmic Pricing Sellers

Now that I have described my methodology, I briefly examine the set of sellers that I find to be doing algorithmic pricing. Table 4.2 shows the number of algorithmic pricing sellers and the number of products they sell that I detect with change thresholds of 10 and 20. In this table, I merge the sellers and products from Crawl1 and Crawl2 and present the total unique numbers.

I immediately observe that many more sellers appear to be using the overall lowest price (and second-lowest price) as the target for their algorithmic pricing than Amazon’s price. However, it is important to note the different strategies are not necessarily mutually exclusive. For example, a seller matches to both Amazon and lowest when Amazon is the lowest price, and a seller matching to the lowest price is likely to often match (i.e., correlate strongly with) the second-lowest price as well. The **Total** row shows the overall unique numbers of sellers and products I detect. In the case when the change threshold is 20, I see that 2.4% of all sellers in my dataset use algorithmic pricing. However, this is 38% of all sellers that have ≥ 20 changes for *at least* one product they sell.

To determine the gap between the prices offered by the suspected algorithmic sellers and the target prices, I plot two figures. Figure 4.12 examines the *absolute* difference between the algorithmic sellers’ prices and the corresponding target prices. I separate sellers matching to the

CHAPTER 4. ALGORITHMIC PRICING ON AMAZON MARKETPLACE

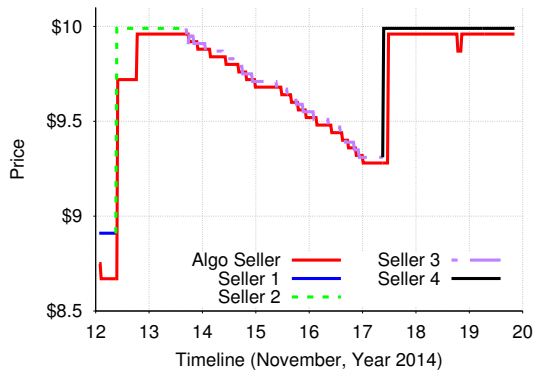


Figure 4.14: Example of 3P seller (in red) matching the lowest price of all other sellers.

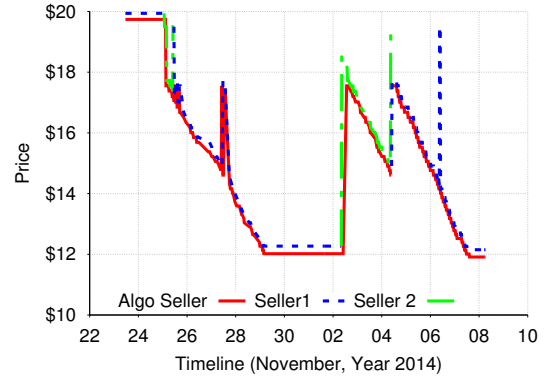


Figure 4.15: A second example of 3P seller (in red) matching the lowest prices offer by two other sellers.

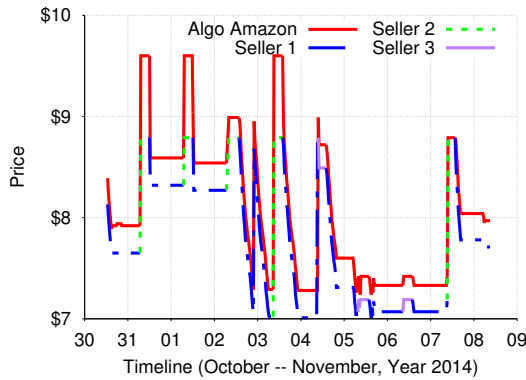


Figure 4.16: Example of Amazon (in red) setting a premium over the lowest price of all other sellers.

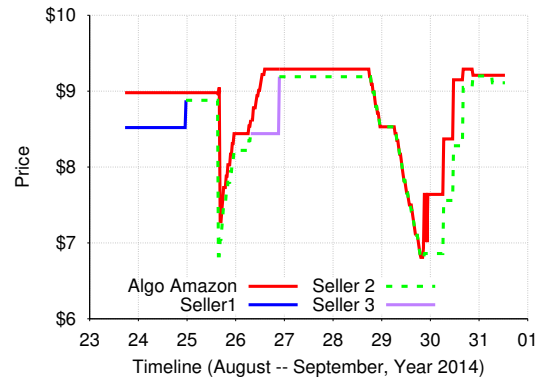


Figure 4.17: Example of Amazon (in red) matching to the lowest price over time.

lowest price and sellers matching Amazon's price in this plot (I ignore the second-lowest price in this plot as matching to the second-lowest price is very similar to matching to the lowest). I observe that algorithmic sellers who match to the lowest price are very close to the lowest price: 70% of these sellers set their price within \$1 of the lowest price. However, only 40% of algorithmic sellers are within \$1 of Amazon's price.

The fact that algorithmic sellers matching to Amazon tend to charge higher prices may be due to the required commission fees that Amazon charges. For example, if a 3P seller and Amazon share the same wholesale cost for a product, the 3P seller must charge a higher price to maintain the same profit margin. As described in § 4.1.1, Amazon's commission fees are around 15% for most

product categories. To see if I can observe algorithmic sellers that include these fees in their prices, I plot the *relative* difference between the algorithmic sellers' prices and the target prices in Figure 4.13. As expected, I see very different behavior between algorithmic sellers matching Amazon's price and the overall lowest price; I can observe a number of sellers who choose a new price that is 15–30% above Amazon's price.

4.4.3 Price Matching Examples

I conclude this section by showing a few example products where I detected algorithmic pricing. First, Figure 4.14 shows an example where a 3P seller has a clear strategy to always match the lowest price across all other sellers. In the figure, I can see four other sellers that offer the lowest price over time, and the algorithmic seller (in red) always quickly matches their price.

Second, I observe several cases where the seller offering the lowest price is able to sell the product well above their reserve price. As shown in Figure 4.15, the algorithmic seller always matches the lowest price from the other two sellers. Although the algorithmic seller is willing to sell the product for as low as \$12, the majority of the time they sell at prices up to 40% higher.

Third, I observe many cases where Amazon itself appears to be employing algorithmic pricing. Figure 4.16 shows a case where Amazon (in red) chooses their price to be a premium above the lowest price of all other sellers. In the figure, I observe that there are three other sellers that offer the lowest price at different points in time, but that Amazon is almost always slightly more expensive.

Fourth, I observe cases that Amazon adopts more complex pricing strategies than simply matching lowest prices. As shown in Figure 4.17, Amazon appears to have a ceiling at around \$9, above which they match the lowest price, but below which they sell the product at a small premium relative to the lowest price.

4.5 Analysis

At this point, I have identified the sellers who are likely using algorithmic pricing. In this section, I compare and contrast the characteristics of algorithmic, non-algorithmic, Prime, and non-Prime sellers. In particular, I am interested in answering the following questions: (1) How do the business practices of algorithmic and Prime sellers compare to non-algorithmic and non-Prime sellers? (2) What fraction of market dynamics are likely caused by algorithmic and Prime sellers? and (3) What is the impact of algorithmic and Prime sellers on the Buy Box? My hypothesis is

CHAPTER 4. ALGORITHMIC PRICING ON AMAZON MARKETPLACE

that Prime sellers will exhibit characteristics which are more similar to algorithmic sellers than non-algorithmic sellers.

Identifying Prime Sellers. I introduce two new classes of sellers in this section: Prime and non-Prime sellers. I label a seller as a Prime seller if it has offered Prime shipping on any product at any epoch in the entirety of my dataset. I find that 70% of Prime sellers offer Prime shipping on all of their products in all epochs, meaning that most Prime sellers exhibit uniform behavior. Although some sellers do not exhibit this uniformity (e.g., they only offer Prime shipping on a subset of their products), these sellers are a minority and I do not expect them to significantly impact my results.

Overall, my dataset includes 7,248 Prime sellers (out of 33,305 total sellers). I observe that 65% of algorithmic sellers are also Prime sellers. This matches my expectation that sellers willing to invest the resources in algorithmic pricing software are also willing to invest in sophisticated logistics capabilities. In contrast, only 5% of Prime sellers are algorithmic sellers, suggesting that the converse relationship is not necessarily true.

4.5.1 Business Practices

To compare the general business practices of sellers, I examine the following four seller-level characteristics: lifespan of products, inventory size, feedback volume, and ranking in the seller page. Note that this list of characteristics is not comprehensive: as mentioned in § 4.3.3, I do not have access to several seller features such as return rate of products, shipping time, *etc.* Since Amazon itself plays a dual role as a merchant and the host of the marketplace, I examine its role as a seller separately.

Product Lifespan. I begin by examining the *lifespan* of seller/product pairs in my dataset. The lifespan of a pair begins the first time I observe a seller offering that product, and ends the last time I observe that seller offering the product. Given my crawling methodology, the shortest possible lifespan is 25 minutes, while the longest are 3 and 1 months for Crawl1 and Crawl2, respectively.

Figure 4.18 shows the distribution of seller/product lifespans. In this and all subsequent figures in this section, I use the same format for presenting results: the “A” line refers to algorithmic sellers, “N-A” refers to non-algorithmic sellers, “P” refers to Prime sellers, and “N-P” refers to non-Prime sellers.

Figure 4.18 shows that algorithmic sellers are active in the marketplace for significantly *longer* periods of time than non-algorithmic sellers. For example, the median seller/product lifetime

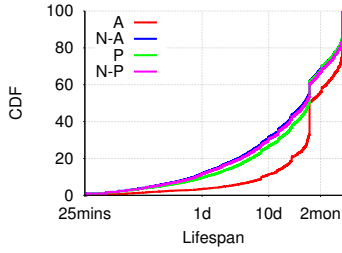


Figure 4.18: Distribution of seller/product lifetimes for different types of sellers.

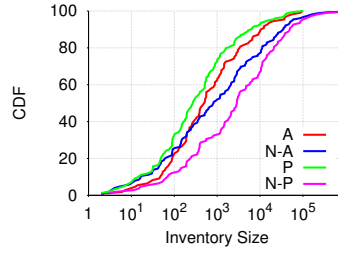


Figure 4.19: Number of unique products offered for sale by different types of sellers.

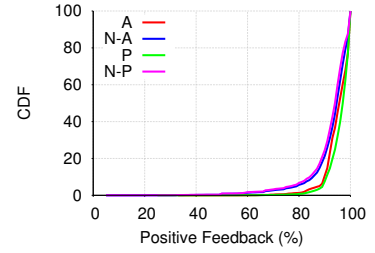


Figure 4.20: Percentage of positive customer feedback received by different types of sellers.

for an algorithmic seller is 30 days, while it is only 15 days for a non-algorithmic seller. Prime and non-Prime sellers closely follow the trend for non-algorithmic sellers lifespans. As I show momentarily, my data suggests that algorithmic sellers have a high sales volume, so the long lifespans of their products further suggest that they have a large amount of inventory. Note that the vertical anomalies in Figure 4.18 around 1 month are artifacts caused by the different lengths of Crawl1 and Crawl2.

Offers, Feedback, and Rank. Next, I compare the total number of products offered for sale by algorithmic and non-algorithmic sellers. Since Crawl1 and Crawl2 focused on best selling products, the dataset may not contain all products sold by sellers. To obtain complete inventories, I conducted a separate crawl that exhaustively collected the entire inventory for 100 randomly selected algorithmic and non-algorithmic sellers, respectively. Note that the inventory for algorithmic sellers includes *all* products they sell, not just specific products where I detect algorithmic pricing.

Surprisingly, as shown in Figure 4.19, algorithmic sellers sell fewer unique products by a large margin versus non-algorithmic sellers. This suggests that algorithmic sellers tend to specialize in a relatively small number of products, perhaps focusing on items that they can obtain in bulk at low wholesale prices. I also see that Prime sellers offer the least number of products for sale, while non-Prime sellers tend to offer the most. This suggests that Prime sellers tend to focus on a limited group of items where they can earn profit while still offering expedient shipping (e.g., it may be difficult for a 3P seller to make a profit offering Prime shipping on a \$5 pencil set).

Next, I examine the feedback received by sellers from customers. On Amazon, customers may rate sellers on a 0–5 scale and also provide feedback about whether their experience with the seller was positive or negative. Amazon presents each seller’s average rating (0–5), percentage of

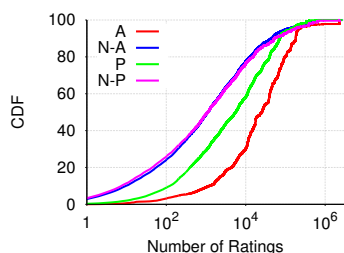


Figure 4.21: Amount of feedback received by different types of sellers.

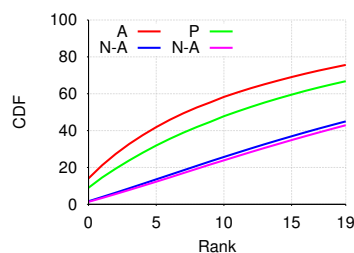


Figure 4.22: Cumulative distribution of rank on the *New Offers* page for different types of sellers.

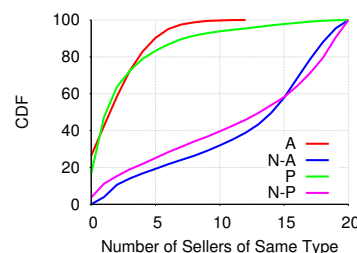


Figure 4.23: Number of sellers of the same type per product.

feedback that is positive (0–100), and total amount of feedback on the *New Offers* pages. Note that Amazon does not display these stats for sellers with insufficient feedback (typically new sellers), and thus I ignore them in the following analysis (this filters out 1.3%, 24%, 6.5%, and 28.4% of algorithmic, non-algorithmic, Prime, and non-Prime sellers in my dataset, respectively).

Figure 4.20 shows the cumulative distribution of positive feedback percentage for all sellers in my dataset. Overall, almost all sellers have greater than 80% positive feedback. While algorithmic sellers garner slightly higher positive feedback than non-algorithmic sellers, the difference is very small. This suggests that algorithmic sellers do not have appreciably better products or customer service versus non-algorithmic sellers.

However, amongst all four seller types, Prime sellers earn the highest overall feedback. One possible explanation for this finding is self-selection: only well-managed, efficient sellers are capable of offering Prime shipping, and these are the same sellers who are likely to have good customer service. An alternative explanation is that Amazon only permits superior sellers to join the Prime program, meaning that Prime sellers must have good feedback by construction.

Figure 4.21 examines the amount of feedback received by different seller types. In this case I observe a stark contrast: algorithmic sellers acquire significantly greater amounts of feedback. There are two reasons why this could be happening: first, algorithmic sellers could have much higher sales volume than other sellers. Second, algorithmic sellers could be more aggressive about soliciting customer feedback. I also observe that Prime sellers receive more feedback than non-Prime sellers.

The feedback and sales volume characteristics of algorithmic and Prime sellers put them at an advantage with respect to non-algorithmic and non-Prime sellers, respectively. As shown in Figure 4.22, when comparing all the products they offer, the rank of algorithmic sellers and Prime

CHAPTER 4. ALGORITHMIC PRICING ON AMAZON MARKETPLACE

sellers on the *New Offers* page tend to be significantly higher than that of non-algorithmic sellers and non-Prime sellers. Note that neither CDF goes to 100% because there are cases where sellers do not appear in the top 20 list.

Taken together, my results show that algorithmic sellers exhibit significantly different characteristics than non-algorithmic sellers: they sell fewer unique products, but in the products where they do compete, they participate in the marketplace for longer periods of time, and they acquire significantly larger amounts of positive feedback (suggesting they may have higher sales volumes). My results also show that Prime sellers usually outperform non-Prime sellers, in terms of quantity and quality of feedback, as well as rank in the seller list. Overall, Prime sellers tend to exhibit similar characteristics to algorithmic sellers, although not always.

4.5.2 Competition

Next, I move on to examining the dynamics of Amazon Marketplace. First, I look into the competition that each seller type faces in the top 20 offers. Next, I look into price changes for each seller/product pair, followed by price and product changes in the Buy Box. Finally, I compare the frequency of different types of sellers winning the Buy Box.

Intra- and Inter-type Competition. I begin by looking at the competition between sellers of the same and different types. Figure 4.23 plots how many sellers of the same type I observe for each product. There are fewer algorithmic and Prime sellers overall in my dataset, and consequently there tend to be fewer algorithmic and Prime sellers per product. For example, 2.6% of products do not have a Prime seller, while 58% have between one and five (inclusive). Similarly, 10.7% of products have no algorithmic sellers, while 72.6% have between one and five (inclusive). In contrast, almost all products are available from a number of non-algorithmic, non-Prime sellers.

The results in Figure 4.23 reveal an interesting facet about the nature of competition in Amazon Marketplace. Amazon's website provides a filter that customers can select to only view products and offers from Prime sellers. Prime members are likely to engage this filter often, as it allows them to quickly find products that are eligible for free two-day shipping. This positions Prime sellers at a significant advantage, since 58% of the time there are at most five Prime sellers for a product, compared to 11 non-Prime sellers. Given that over half of Amazon's customer base are now Prime members [166], this effectively means that Prime sellers have much less competition with respect to a large and lucrative segment of Amazon's customer base.

Figure 4.23 also supports my arguments explaining some of the observed differences

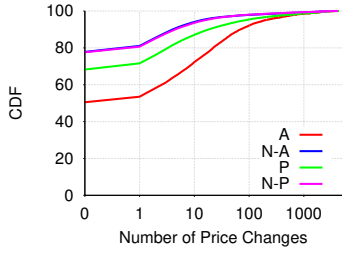


Figure 4.24: Number of price changes per seller/product pair.

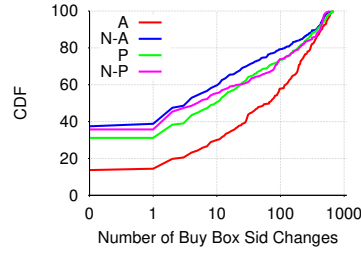


Figure 4.25: Number of changes in the Buy Box seller for products with and without algorithmic and Prime sellers.

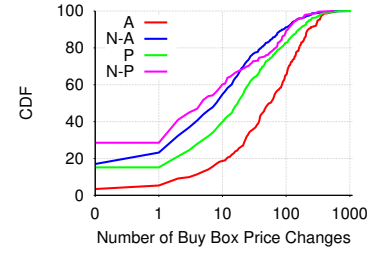


Figure 4.26: Number of changes in the Buy Box price for products with and without algorithmic and Prime sellers.

between Prime sellers and algorithmic sellers. Prime sellers do not need to compete with the entire space of algorithmic sellers, only with the 5% of Prime sellers who *also* use algorithmic pricing. Since such a small fraction of their competition uses algorithmic pricing, it is possible that some Prime sellers are not incentivized to employ algorithmic pricing techniques because it does not confer an appreciable competitive advantage.

Price Changes. Next, Figure 4.24 shows the distribution of number of price changes per seller/product pair. I observe that for non-algorithmic and non-Prime sellers, the price *never* changes for 71% of seller/product pairs. This suggests that either there is no competitive pressure to instigate changes, or these sellers are simply unsophisticated and unresponsive.

However, if I examine the algorithmic sellers, I observe a very different distribution. Seller/product pairs with at least one price change are more common than not,¹⁰ and there is a long tail of products whose prices change 100 or even 1000 times. I also observe that Prime sellers frequently change prices, although not as much as algorithmic sellers.

Buy Box. Next, I examine the impact of algorithmic sellers and Prime sellers on the Buy Box. Figures 4.25 and 4.26 compare the number of seller and price changes I observe in Buy Boxes for products that have algorithmic sellers and Prime sellers, and products that do not. As expected, products with algorithmic sellers experience many more price and seller changes in the Buy Box: for example, 20% of products without algorithmic sellers have zero price changes, versus only 2% for products with algorithmic sellers. Unfortunately, this exposes customers to a great deal

¹⁰Readers may notice that the number of price changes for algorithmic sellers in Figure 4.24 goes below 20, which is the change threshold I set to detect those algorithmic sellers. However, this plot includes *all* the products sold by the algorithmic sellers, some of which have <20 changes during my crawl.

CHAPTER 4. ALGORITHMIC PRICING ON AMAZON MARKETPLACE

of volatility, which they may perceive to be confusing and undesirable [18]. On the other hand, Prime sellers have a less disturbing effect on the Buy Box. This concurs with my observation that only 5% of Prime sellers use algorithmic pricing, thus facilitating more stable prices.

Although products with algorithmic sellers tend to have more dynamic Buy Boxes, this does not necessarily mean that algorithmic sellers are more likely to actually win the Buy Box. To assess this, I examine which types of sellers are more successful at winning the Buy Box in Figure 4.27. I find that algorithmic sellers are indeed more likely to win the Buy Box at all ranks except for the top one, which is won more by non-algorithmic and Prime sellers. Given the importance of winning the Buy Box, this result is quite interesting: as shown in Figures 4.12 and 4.13, algorithmic sellers tend to set their prices greater than or equal to the lowest price for a product. However, even though algorithmic sellers do not offer the lowest prices, they manage to win the Buy Box anyway due to other attributes like positive feedback.

4.5.3 Amazon as a Seller

Given Amazon's dual role as merchant and host of the marketplace, it makes sense to examine their role as a seller separately. Table 4.3 shows the percentage of best-selling products that are sold by Amazon. "Overall" means that Amazon sold the product at some point during my crawls, while "Each Crawl" is the average percentage of products sold by Amazon during each snapshot. Although Amazon's for-sale offerings do change over time, they still dominate the market, offering greater than 72% of all best-selling products at any given time.

Besides offering an enormous number of products, Amazon is also known for their low prices. To demonstrate this, I plot Figure 4.28, which shows the the rank of Amazon as a seller on the *New Offers* page for every product in each crawl. For this analysis, I divide products into four categories: 1) products with at least one algorithmic seller other than Amazon; 2) products where Amazon is the only algorithmic seller; 3) products with at least one Prime seller other than Amazon; and 4) products where Amazon is the only Prime seller. Note that categories [1, 2] and [3, 4] are mutually exclusive, but other pairings are not. Also recall that, as I observed in Figure 4.7, the seller list is *roughly* ordered from least-to-highest price.

Overall, I observe that Amazon ranks highly for almost all products they sell. However, if I compare the products that have sellers doing algorithmic pricing to products without such sellers, I see a dramatic difference. In products with algorithmic sellers, I see that Amazon ranks in the top 5 sellers 90% of the time; in products without such sellers, I see this percentage increases to 97%.

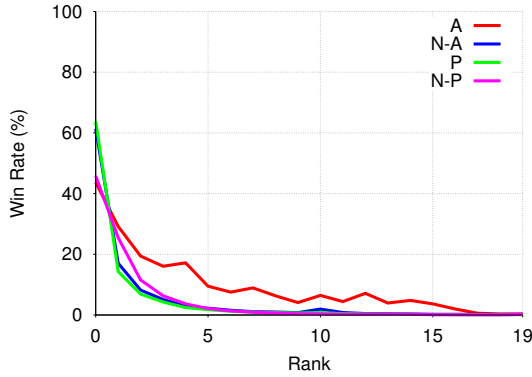


Figure 4.27: Probability of winning the Buy Box for different types of sellers at different ranks.

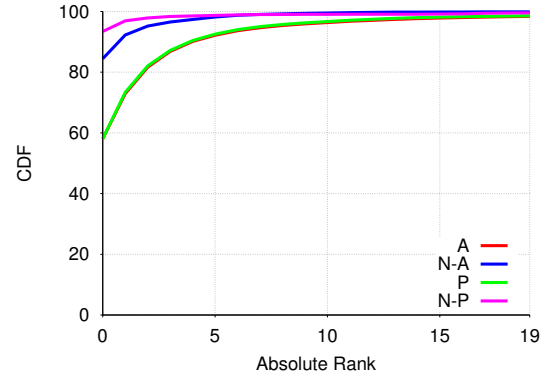


Figure 4.28: Cumulative distribution of Amazon's rank in the presence/absence of algorithmic and Prime sellers.

Prime sellers are as effective as algorithmic sellers in disrupting Amazon's contention for the top rank. Thus, even Amazon tends to be ranked lower for products where other algorithmic sellers and Prime sellers are active. Non-Prime sellers have the least impact on Amazon's rank.

4.6 Simulations

So far, I have successfully classified sellers into two discrete types: algorithmic sellers and non-algorithmic sellers. I then compared these two types of sellers on various explicit metrics in § 4.5, such as inventory size, lifespan, feedback, *etc.*

In this section, I quantify a seller's performance on Amazon Marketplace using two metrics: *checkouts* and revenue. Since I cannot measure these quantities directly, I use simulations that are driven by my real-world data to estimate them. I begin by describing the methodology behind my simulations, and then analyze results. I find that the distribution of revenue is highly skewed in favor of algorithmic sellers, and that Amazon in particular is in a class of its own.

I conclude the section by using simulations to examine what it would take for sellers who

	Algorithmic	Non-Algorithmic
Overall	72%	73%
Each Crawl	62%	63%

Table 4.3: Percentage of products with Amazon as one of the listed sellers.

CHAPTER 4. ALGORITHMIC PRICING ON AMAZON MARKETPLACE

do not win the Buy Box in my real-world data to “change their fate” and win the Buy Box. This simulation is motivated by my observation that winning the Buy Box is crucial for sellers to obtain checkouts and revenue. Unfortunately, I find that most sellers cannot win the Buy Box even if they try, and that the minority of sellers who could potentially win the Buy Box would need to lower their prices drastically in order to do so.

4.6.1 Methodology

I begin by attempting to simulate the number of checkouts and total revenue earned by sellers on Amazon Marketplace. For the purposes of this discussion, I define a “checkout” to mean that a customer adds a specific product offered by a specific seller to their shopping cart and then purchases it. Revenue simply refers to the amount in dollars earned by a seller when a given product is checked out. Note that each seller may have their own list price for a given product (which I draw from my crawled data), and that revenue per product includes any associated shipping costs (since Amazon’s interface lumps product price and shipping cost together). Furthermore, I make no attempt to subtract Amazon’s listing fees, commissions, *etc.* from the revenue earned by third-party sellers.

I examine checkouts and revenue separately because sellers and products have vastly different properties on Amazon Marketplace. For example, a seller that offers 1,000 products may amass more checkouts than a seller that only offers one product, simply due to greater scale. However, the seller with one product could potentially earn more revenue, if that single product is an extremely high-priced item. Rather than try to develop a single metric that takes all of these factors into account, I opt to examine checkouts and revenue separately.

Unfortunately, my crawled data does not include data about product purchases, so I cannot directly measure checkouts or revenue. Instead, I estimate the number of checkouts and revenue that each seller will amass over time by simulating the behavior of customers. Specifically, I assume that in each time epoch (i.e., 25 minute interval), each product in my crawled data is visited and purchased by one customer. In other words, I assume *each product in my dataset is purchased at a constant rate over time*, and that *the rate for all products is equal*. Although I could assume more than one purchase per epoch per product, this would not impact the results of the simulation since the increased purchase rates would uniformly impact all sellers.

Buyer Strategies. The vast majority of products in my dataset are offered by multiple sellers, which raises an important question: *during each epoch, which seller does my simulated customer purchase from?* Since I do not know what customers’ actual purchasing strategies are, I

CHAPTER 4. ALGORITHMIC PRICING ON AMAZON MARKETPLACE

simulate four different strategies:

1. *Always Lowest*: The customer always purchases the product from the seller with the lowest price.
2. *Always Buy Box*: The customer always purchases the product from the seller in the Buy Box.
3. *82/18 Lowest*: The customer purchases the product from the seller in the Buy Box with 0.82 probability, and from the lowest priced seller with 0.18 probability. Note that the lowest priced seller may also be the occupant of the Buy Box.
4. *82/18 Weighted*: The customer purchases the product from the seller in the Buy Box with 0.82 probability. The remaining 0.18 probability is divided amongst all sellers in a weighted manner based on price sorted from lowest to highest, i.e., the lowest price seller receives the most probability, the second lowest receives the second most probability, and so on.

The “Always Lowest” and “Always Buy Box” strategies represent very simple, straightforward models of customer behavior. In contrast, the “82/18” models are more realistic: available evidence suggests that customers on Amazon purchase from the seller in the Buy Box 82% of the time [177], which motivates my models. Arguably, the “82/18 Weighted” strategy is most realistic, since it takes evidence of real Buy Box behavior into account [177] and simulates a pseudo-rational customer that prefers to save money, but may pay higher prices for exogenous reasons (e.g., faster shipping times or better seller feedback scores).

Execution. To execute my simulations, I rely on my crawled data. For each product in each epoch, I calculate 1) the expected checkout probability for each seller and 2) the expected revenue for each seller. Expected revenue for a seller is simply their offer price for the item times their checkout probability. I iterate over all product epochs and sum the checkout probabilities and revenue for each seller. Finally, I normalize the checkout probability for each seller by dividing it by that seller's total number of product epochs (i.e., I calculate the average checkout probability for each seller). I execute this process once for each of my customer strategies, for a total of four simulations. Overall, my simulations cover more than 1.39 million product epochs.

Limitations. Admittedly, my customer behavior models are simplistic. First, as noted above, I make the simplifying assumption that all products in my dataset are purchased at equivalent rates. This is clearly untrue: Amazon presents a “Best Sellers Rank” for many products on their website that reveals how well the product sells relative to other similar products. Unfortunately, this

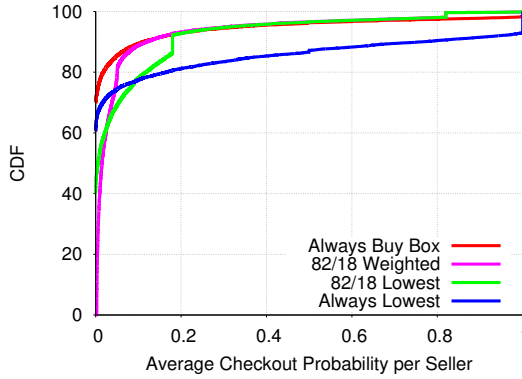


Figure 4.29: Percentage of checkouts amassed by each seller under four different customer behavior models.

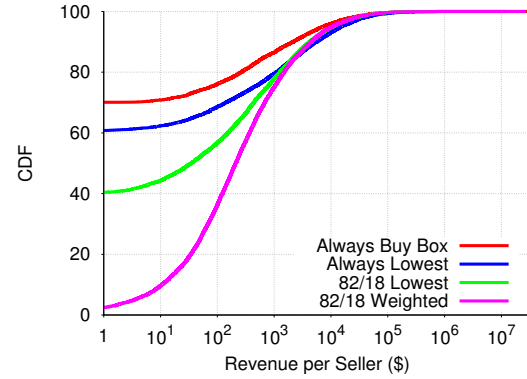


Figure 4.30: Expected revenue earned by each seller under four different customer behavior models.

data is too coarse to use for refining my simulations, since I cannot use it to compare the sales of products in different categories (e.g., a book versus a shovel), and the ordinal ranks do not reveal the actual sales gaps between products (i.e., it is not clear how much more a #1 rank product sells than a #2 rank product).

Second, I assume that the purchase rates for products are constant over time. Again, this is a simplifying assumption: it is likely that products sell more rapidly during daylight hours when people are awake, and perhaps on weekends when they have more time to browse Amazon Marketplace. I do not have any available data to help me quantify these effects, and thus I make no attempt to incorporate them into my models.

Ultimately, the simulation results I present in this section are meant to highlight broad trends on Amazon Marketplace, including the relative advantages gained by algorithmic sellers, and the dominant position of Amazon itself. I make no claims to specificity, i.e., the number of checkouts and revenue that I estimate for a given seller almost certainly *do not* represent the precise quantities earned by the seller in reality.

4.6.2 Simulation Results

Figures 4.29 and 4.30 show the distributions of checkouts and revenue per seller, respectively, in my simulations. Each line shows the distribution under a different model of customer behavior.

The first interesting feature of Figures 4.29 and 4.30 is the number of sellers who receive

CHAPTER 4. ALGORITHMIC PRICING ON AMAZON MARKETPLACE

zero checkouts and revenue. By design, this quantity is highly model-dependent. The “Always Lowest”, “Always Buy Box”, and “82/18 Lowest” models are highly restrictive, since most sellers in my dataset never obtain the Buy Box or offer the lowest price for a product. Under the “Always Buy Box” model, 70% of sellers amass zero clicks and revenue, while under the slightly-less restrictive “Always Lowest” model 61% of sellers amass zero clicks and revenue. In contrast, under the “82/18 Weighted” model, all sellers have some probability of being selected by a customer (even if those probabilities are very small for the majority of sellers).

A second important observation from Figures 4.29 and 4.30 is how quickly the models converge. With respect to checkouts, in three of the models $\sim 93\%$ of sellers have ≤ 0.2 probability of acquiring checkouts (on average). The exception is the “Always Lowest” model, in which 80% of sellers have ≤ 0.2 probability of acquiring checkouts. Similarly, $\sim 95\%$ of sellers earn \$10,000 or less, irrespective of the chosen model.

The final interesting facet of Figures 4.29 and 4.30 is how dramatically checkouts and revenue are skewed towards a small subset of sellers. For example, 78–87% of sellers earn less than \$1,000 in revenue over the course of my simulation, depending on the customer model. The top three sellers in terms of earnings each make more than the bottom 87% of sellers *combined*. Amazon itself earns the most revenue in all of my simulations by two orders of magnitude, due the fact that it offers more products for sale than any other seller, plus it can charge more per product than competitors and yet still win the Buy Box in a majority of cases.

I caution that the results in Figure 4.30 do not necessarily mean that the majority of sellers on Amazon are not profitable. I do not know seller’s profit margins, fixed overhead costs, or sales volumes, and thus cannot draw conclusions about profitability. Rather, the results in Figure 4.30 simply highlight the disparity in earnings potential between sellers on Amazon Marketplace.

Checkouts. Next, I take a closer look at checkouts per seller by separating algorithmic from non-algorithmic sellers. Figure 4.31 shows the percentage of checkouts per seller for algorithmic and non-algorithmic sellers under all four of my customer purchase models. In these plots, each dot is a different seller; I highlight Amazon using an “ \times ”.

Under three of the strategies (excepting “Always Lowest”), algorithmic sellers amass more checkouts than non-algorithmic sellers. In contrast, Figure 4.31a (the “Always Lowest” model) reinforces my finding that algorithmic sellers do not necessarily compete on price. Instead, algorithmic sellers appear to be exploiting the behavior of Amazon users to their advantage: since 82% of users purchase directly from the Buy Box, algorithmic sellers focus on winning the Buy Box without

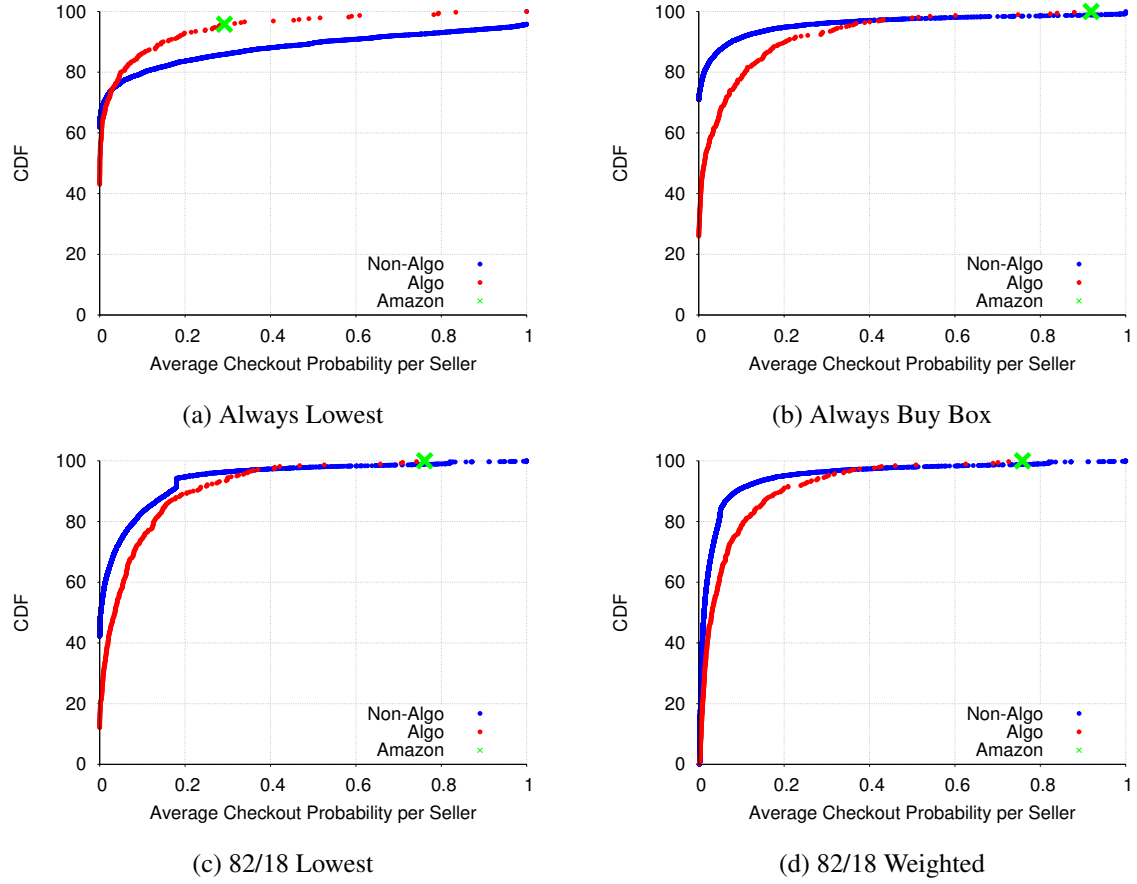


Figure 4.31: Percentage of checkouts per seller for algorithmic and non-algorithmic sellers, under four different models of customer behavior.

always offering the lowest prices. Recall that in Figures 4.12 and 4.13, I observe that algorithmic sellers tend to set prices that are higher than the target price for a given product, and yet as seen in Figure 4.27, algorithmic sellers are more adept at winning the Buy Box than other sellers. This nuance is best exemplified by Amazon itself, who has 0.9 probability (on average) of acquiring checkouts under the “Always Buy Box” model, but only 0.3 probability under the “Always Lowest” model. Overall, I observe that the relative advantage held by algorithmic sellers is largest under the “Always Buy Box” model, and less so under the “82/18” models.

Revenue. Next, I examine revenue per seller for algorithmic and non-algorithmic sellers in Figure 4.32. Unsurprisingly, algorithmic sellers tend to earn at least an order of magnitude more revenue than non-algorithmic sellers, due to the greater number of checkouts they amass. This observation is even true under the “Always Lowest” model (Figure 4.32a), indicating that algorithmic

CHAPTER 4. ALGORITHMIC PRICING ON AMAZON MARKETPLACE

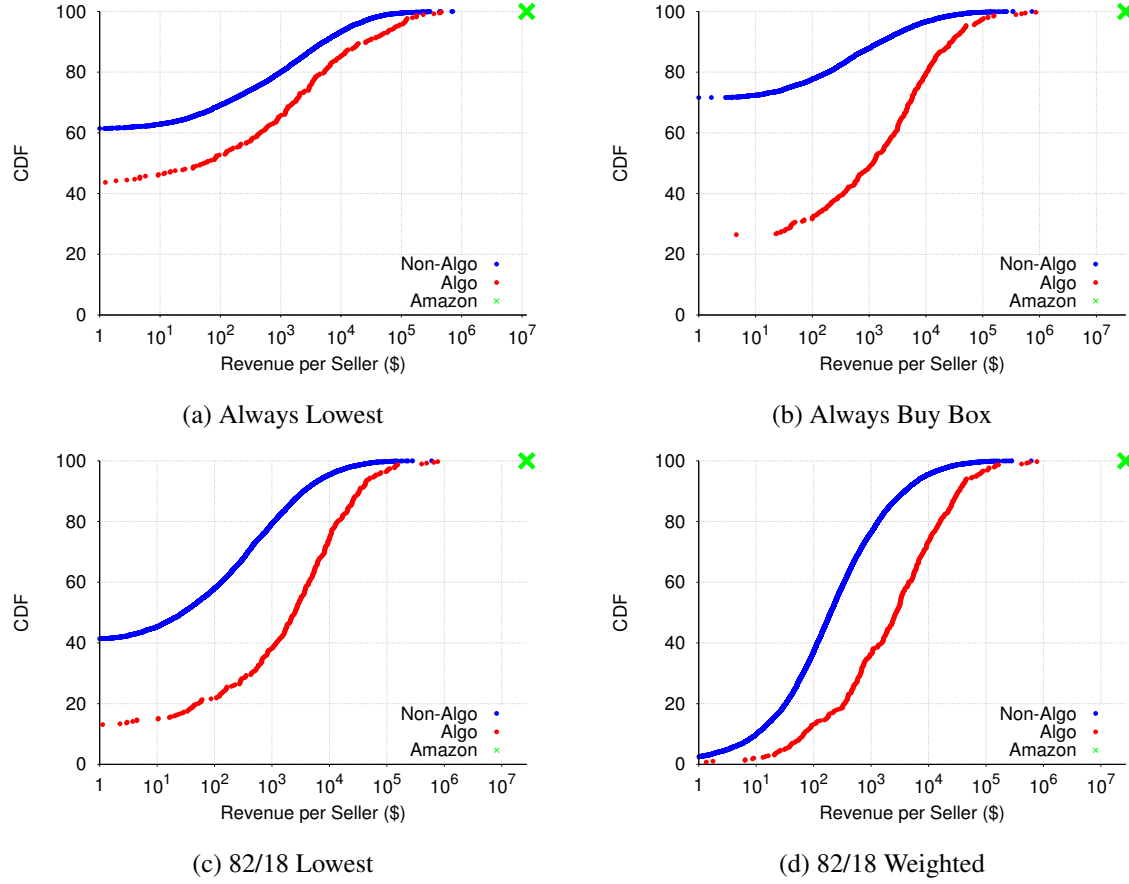


Figure 4.32: Revenue per seller for algorithmic and non-algorithmic sellers, under four different models of customer behavior.

sellers still earn more revenue even with relatively fewer checkouts.

By far, Amazon itself earns the most revenue, regardless of customer purchasing model. As I note earlier, this is due to a combination of factors, including the large number of products they offer, relatively higher prices than competitors, and the ability to consistently win the Buy Box.

My simulation results highlight the impact of Amazon’s design choices when constructing their marketplace. Recall that Amazon’s probability of acquiring checkouts reduces from 0.9 under the “Always Buy Box” model to 0.3 under the “Always Lowest” model. This translates to a \$20 million reduction in revenue for Amazon in my simulations. In other words, if Amazon were to always place the lowest priced offer into the Buy Box (instead of their current opaque algorithm), they would lose a large amount of revenue¹¹. This might force Amazon to change their pricing

¹¹Note that in my simulations, I do include commissions from sales by third-party sellers in Amazon’s revenue total.

strategies, by lowering their offer prices and accepting less revenue per item, but more revenue overall through increased sales volume.

4.6.3 Breaking Into the Buy Box

With the majority of buyers purchasing directly from the Buy Box, it is imperative for every seller to win the Buy Box as frequently as possible. However, as I show, the Buy Box tends to be dominated by a subset of sellers who are consequently able to capture a disproportionately large fraction of all checkouts and revenue. This raises an interesting hypothetical question: *what would it take for sellers who are currently losing the Buy Box to win it?*

In this section, I investigate if it is possible for a seller to win the Buy Box for a given product by changing their offer price (and if so, by how much their offer price must change). Recall that in § 4.3, I find that some of the features that influence the Buy Box algorithm are not directly under sellers' control. For example, a seller cannot choose to suddenly have large quantities of high-quality feedback. However, product price is the most important feature for the Buy Box algorithm, and prices are directly set by sellers.

Sniping. To investigate the impact of price changes on the Buy Box, I again use data-driven simulations. Specifically, I assume that a seller wants to win the Buy Box for a specific product, so they change their price right before Amazon calculates the Buy Box occupant for the subsequent time epoch. I refer to this practice as “sniping”, since it does not give other sellers the opportunity to react to the sudden price change before the new Buy Box occupant is chosen. Sniping makes my simulation tractable, since otherwise I would also need to account for the reactions of other (algorithmic) sellers in response to the change of the product's target price.

Based on this scenario, I conduct simulations using the following two-step methodology. *First*, I train a linear logistic regression classifier on all the seller features listed in § 4.3. The classifier learns a decision boundary that best separates sellers who win the Buy Box from sellers who lose the Buy Box. *Second*, I try to determine the new price at which each losing seller can win the Buy Box for each product epoch in my dataset. To win the Buy Box, a losing seller must change their price such that it places them at a greater positive distance to the decision boundary than the existing Buy Box winner (i.e., the winner I observe in my data). I determine the new “winning” price for each

Thus, although Amazon may lose revenue from direct sales, they may make up the shortfall through higher commission revenue.

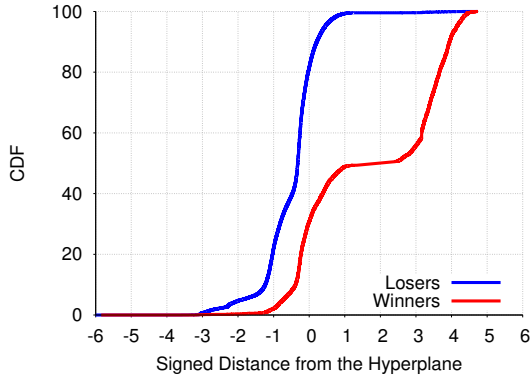


Figure 4.33: Distribution of sellers across the hyperplane in my logistic regression classifier.

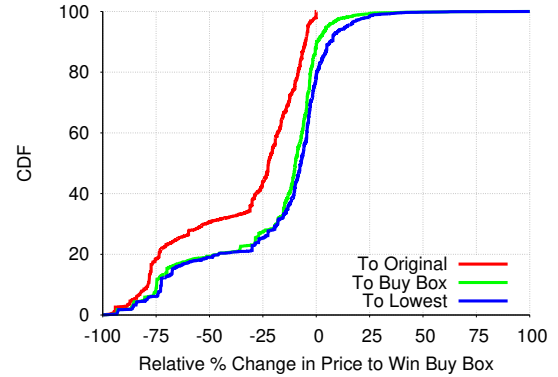


Figure 4.34: Price change required for sellers to win the Buy Box, relative to three baselines.

seller by running a binary search between \$0.01 (i.e., the minimum amount that a product can be listed for) and the original price offered by the seller.

Validation. The logistic regression classifier achieves 61% accuracy for *predicting* the ground-truth Buy Box winners¹². Figure 4.33 shows the signed distance to the hyperplane for all sellers in all epochs in my dataset. I observe some false negatives, i.e., sellers that won the Buy Box but are predicted to have negative distance, and false positives, i.e., sellers that lost but are predicted to have positive distance.

To mitigate the impact of false positives and negatives on my simulations, I only simulate price changes in product epochs where the classifier correctly predicts the outcome for *all* sellers. This filter leaves me with 111,932 product epochs to analyze.

Results. In total, I run the sniping simulation for 1.5 million $< \text{product}, \text{seller}, \text{epoch} >$ triples, which covers 5,822 unique sellers. This includes 257 algorithmic sellers (out of 543 total).

Of the 5,822 sellers that attempt to snipe the Buy Box, a mere 883 were successful at least once. These 883 sellers included 107 algorithmic sellers. This result paints a stark picture of the competitive landscape on Amazon Marketplace: even if many sellers drop their prices to \$0.01, they still cannot win the Buy Box due to factors that may be beyond their control.

Figure 4.34 plots the price changes (relative to different baselines) that the 883 successful

¹²For 61% of all product epochs, the classifier returns the highest probability for the correct Buy Box winner. While the classifier's accuracy for *classifying* Buy Box winners and losers is 69% and 81% respectively, these numbers are misleading since two (or more) sellers can be classified as Buy Box winners for the same epoch.

CHAPTER 4. ALGORITHMIC PRICING ON AMAZON MARKETPLACE

snipers had to incur to win the Buy Box. In 56% of successful sniping attempts, the seller would need to decrease their offer price by 25% or less relative to their original offer price. This corresponds to an 11% price reduction relative to the current Buy Box occupant's price. Note that in 11% of cases, the sniping seller can actually *increase* their offer price above the Buy Box occupant's price to win the Buy Box, which suggests that there are opportunities for some sellers to engage in more aggressive dynamic pricing tactics.

Whether the sniping strategy is viable for sellers in practice is unknown. I do not have data on seller's profit margins, and it is unclear whether each seller's business would be sustainable if they were forced to accept large price reductions in exchange for winning the Buy Box. In some cases, I find that sellers would need to cut their prices by close to 100%, which would clearly lead to selling products at below-cost.

4.7 Summary

In this chapter, I took the first steps towards detecting and quantifying sellers using algorithmic pricing on Amazon Marketplace. I collected large-scale data on products and sellers on Amazon Marketplace, and I make my code and data available to the research community.¹³ I found that algorithmic sellers using a specific range of strategies can be detected using a target price time series, and I identify over 500 such sellers in my data set. I observe cases where algorithmic sellers change prices tens or even hundreds of times per day, which would be difficult for a human to maintain over time—especially one attempting to manage many products simultaneously—but is trivially automated.

My findings illustrate the power of algorithmic pricing in online marketplaces. Sellers I identified as using algorithmic pricing receive more feedback and win the Buy Box more frequently. My data-driven simulations suggest that algorithmic sellers have higher sales volumes and thus more revenue than non-algorithmic sellers, under a variety of different customer purchasing models. Clearly, the existence of cost-effective, user-friendly automation platforms like Sellery and Feedvisor is a win for sellers, especially smaller merchants who lack a dedicated programming staff.

However, the introduction of algorithmic pricing makes the sales competition extremely challenging for non-algorithmic sellers. In addition, automating price changes opens the door to intentional and unintentional market distortions. Finally, high pricing dynamics may confuse customers during online shopping activities.

¹³Available at <http://personalization.ccs.neu.edu>

Chapter 5

Impact of Gender on Hiring Sites

The main goal of this study is to investigate if ranking bias or unfairness, with respect to gender, exist in the ranking algorithms used by major online hiring websites. Furthermore, I explore whether there exist alternative ranking methods that can mitigate the inequalities present in search results on these websites.

I begin this chapter by introducing the websites covered in this study, *resume search engines* in general, and the scope of this work in §5.1. In §5.2, I describe my data collection methodology, and the specific candidate-level variables I extract from the data. In §5.3, I explore the *ranking bias* and *unfairness* exhibited by the ranking algorithms, and investigate whether these issues are caused by explicit or implicit use of gender as a feature. Based on my findings, I propose two alternative ranking methods that each obey a different fairness constraint in §5.4. Finally, I summarize this study in §5.5.

5.1 Background

In this section, I discuss search engines in general, and the three specific websites that are the focus of my study. I begin by discussing *resume search engines* at a high-level, and which specific aspects of the information retrieval process I am able to examine in this study. Next, I present both the candidates' and the recruiters' perspective of each hiring website to highlight the similarities between the websites, as well as salient differences in their resume search engines.

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

The screenshot shows the Indeed website interface. At the top, the search criteria are 'software engineer' (under 'what:') and 'New York, NY' (under 'where:'). Below these, it says '48,783 resumes' and 'Find Resumes' with a link to 'Advanced Search'. On the left sidebar, there are links for 'Saved' and 'Contacted'. Below these, it says 'software engineer resumes in New York, NY'. Further down, there are filters for 'Sort by: relevance - date', 'Distance: within 25 miles', and 'Last Updated: show all resumes'. The main content area shows a list of resumes. The first two are highlighted with orange arrows and text: 'Select up to 50 resumes to contact'. The first resume is for a 'Software Test Engineer' at 'Lodestone Software Services Pvt. Ltd' in 'Jersey City, NJ', updated 'Mar 18'. The second resume is for a 'Software Engineer' at 'Sage Payroll Services' in 'New Brunswick, NJ', updated 'Mar 16'.

(a) Indeed

The screenshot shows the Monster website interface. At the top, it says 'Search Results (1-20 of 1000 Candidates)' with links for 'Too few results?' and 'Too many results?'. Below this, there's a 'Browse Candidates (50 Pages)' section with a 'Select' dropdown, '0 candidates selected', and navigation buttons. The main content area shows a list of candidates. The first candidate is a 'Senior Software Engineer, SiTAPP Moscow Russia' with a '10.0 match' score. They have '20.2 yrs' of 'Software Engineering' experience and '12.2 yrs' of 'Eclipse IDE' and 'Java' experience. The second candidate is a 'Senior Software Engineer, Alcatel-Lucent' with a '10.0 match' score, '20.9 yrs' of experience, and '12.2 yrs' of 'Eclipse IDE' and 'Java' experience. Both candidates have 'Resume Updated' 20 months ago and 'US Authorized' status.

(b) Monster

Figure 5.1: Screenshots of search results on Indeed and Monster when searching for a “Software Engineer” near New York City.

5.1.1 Resume Search Engines

In this study, I examine the *resume search engines* provided to recruiters by major hiring websites. Before I describe the specific hiring sites that I target in this study, it is instructive to first describe these search engines at a high-level, as this impacts what I am able to investigate.

Information Retrieval (IR) systems are designed to return *relevant* information from a corpus of data in response to *queries* from users. Perhaps the most familiar example is Google Search, which accepts free-text queries and returns links to websites from a crawled corpus of webpages.¹ The kinds of queries that a system accepts, the contents of the corpus, and the definition of relevancy all vary, depending on any given system's context.

Here, I examine resume search engines, which are designed for recruiters who are looking to identify and recruit new employees. In this case, the corpus contains resumes and personal profiles uploaded by prospective candidates seeking employment. Recruiters query by entering a free-text job title, but they may also add *filters* to further refine their search, e.g., years of experience, minimum educational level, *etc.* The resume search engine returns a ranked lists of candidates that the system judges to be relevant to the queried job title, subject to any specified filters. Figure 5.1 shows examples of search results from Indeed and Monster.

Behind the scenes, three processes determine the composition and order of search results produced by IR systems.

1. *The items in the corpus* (i.e., candidates and resumes) restrict what results can possibly be returned by the system.
2. *The relevancy metric* determines which items the IR system judges are relevant to a given query (i.e., job title).
3. *The ranking algorithm* determines the ordering of the relevant items that are displayed to the user (i.e., a recruiter).

In practice, process (2) and (3) may be implemented by a single algorithm: all items in the corpus are ranked in response to a query, and the top- k are returned to the user in that order. However, conceptually, determining relevancy and order can be separated.

My purpose in this study is to examine the results returned by resume search engines through the lens of gender. For example, *do female candidates tend to be ranked lower than equally qualified male candidates?* In practice, all three of the processes outlined above can impact the

¹This is a gross simplification of how Google Search actually works, but it is still useful as a high-level example.

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

composition and order of search results with respect to gender [114]. The population in the corpus determines how many men and women are available overall, so if, for instance, women use hiring websites less frequently than men for some exogenous reason, then the search results may end up being primarily composed of men. Similarly, if the relevancy metric or ranking algorithm heavily weights features that are associated with male candidates, then the search results may include a disproportionately large number of male candidates, and/or show men at higher ranks.

Scope of my Work. In this study, I focus on process (3), the ranking of candidates. Although process (1) and (2) are interesting, I cannot critically examine them because there is no way for me to enumerate all candidates in a hiring website's corpus. Without this information as a baseline, I cannot investigate things like gender skew in the relevancy metric (i.e., returning more candidates of a specific gender than would be expected given the overall composition of the population in the corpus). As I describe below, what I can crawl are search results, which does enable me to investigate the relationship between gender and rank. Thus, I do not focus on the composition of search results (i.e., which candidates are deemed relevant), but on ranking (i.e., how are the relevant candidates ordered).

5.1.2 Hiring Websites

I chose three hiring sites to examine in this study: Indeed, Monster, and CareerBuilder. I chose these sites for several reasons. *First*, they are three of the most popular employment websites in the U.S., according to the Alexa ranking [17] (along with LinkedIn and Glassdoor). Indeed is the second highest-traffic job search site in the U.S. behind LinkedIn. It serves 200 million unique visitors per month and 16 million job postings [101]. Monster claims to have 29 resume uploads, 7900 job searches, and 2800 job views per minute [133], while CareerBuilder claims to have millions of candidate visits per month and job listings from thousands of employers [45].

Second, all three sites provide economical access to their resume search engine. Indeed does not charge for access, while Monster and CareerBuilder charge \$700 and \$400 per month, respectively (as of early 2016). This made it feasible for me to sign-up and collect data from these sites. Contrast this to LinkedIn, which charges \$9000 for access to its unrestricted recruiter tools.

Third, as I discuss next, all three sites have similar user interfaces for candidates and recruiters. This makes the sites roughly comparable in terms of usability and features, which allows me to contrast the output of the resume search engines across the sites.

Candidate's Perspective. The three sites are very similar from a candidate's perspec-

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

tive. Before searching and applying for jobs, candidates must register for a free account, and possibly fill out a personal profile and upload a resume. The amount of profile information that is mandatory varies across the sites; on Monster, users must provide their name, location, educational attainment, and previous job. In contrast, CareerBuilder allows users to leave their profile empty. However, all three sites continuously remind users to upload more information, especially a resume, since the job recommendation functionality on the sites depends on this information.

Once a candidate has created a profile, the sites give them two options for finding open positions: keyword search and recommendations. All three sites have *job search engines*, and provide filters for narrowing down the results (e.g., by location, skill requirements, *etc.*). The sites will also recommend positions to candidates based on their profile and resume information, although the specific algorithms used to generate recommendations are unknown. On all three sites, candidates may apply to as many jobs as they wish.

Recruiter's Perspective. At a high-level, all three sites offer similar capabilities for companies and recruiters. Recruiters may post open positions and inspect candidates that apply. Monster and CareerBuilder charge recruiters a per-listing fee to post jobs. Postings on Indeed are free, but recruiters may “sponsor” their postings for a fee, which increases their rank in job search results.

The three hiring sites also offer *resume search engines* that recruiters can use to proactively identify candidates. Recruiters may specify a job title and filters, and the search engines return ranked lists of candidates that match the given criteria. **None of the sites allow recruiters to filter or order search results by demographics** (e.g., gender, ethnicity), but proxy variables exist in some cases (e.g., years of experience as an indicator of age). By default, the search results are sorted by opaque metrics (e.g., “most relevant”), although the recruiter may re-sort the list by objective metrics like years of experience. Figure 5.1 shows the search interface on Indeed and Monster; the query term, location, and filters are all shown in the left column of the page. CareerBuilder has a very similar layout.

Search Result Format. Since the goal of my study is to examine the order of candidates returned by resume search engines, I must discuss the specific implementation of resume search on each hiring site. On all three sites, a recruiter must supply a job title and a geolocation to initiate a search. By default, the search engine returns all relevant candidates within 25–50 miles of the specified location.

On Indeed, resume search results display 50 candidates per page, up-to a maximum of

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

Origin	Feature	Description	Present On		
			I	M	C
Observed in Search Results	Job Title Relevance	Relevance of the searched job title to the candidate's current job title	✓	✓	✓
	Skills Relevance	Relevance of the searched job title to the candidate's skills	✓	✓	×
	Education Level	Education level of the candidate	✓	✓	✓
	Job Popularity	Popularity of the candidate's current job title among all candidates returned for the searched job title	✓	✓	✓
	Last Modified	The recency of the candidate's profile and resume	✓	✓	✓
	Experience	The experience of the candidate in years	✓	✓	✓
	Relocate	Whether the candidate is willing to relocate (binary)	×	✓	×
	Skills Popularity	Popularity of the candidate's skills among all candidates returned for the searched job title	×	✓	×
	Information Relevance	Relevance of the searched job title to additional profile info	✓	×	×
	Bio Relevance	Relevance of the searched job title to the candidate provided description of the working experience	✓	×	×
	Skills Match	Whether the entire searched job title is present in the candidate's skill set (binary)	✓	×	×
	Information Match	Whether the entire searched job title is present in the candidate's additional profile information	✓	×	×
	Bio Match	Whether the entire searched job title is present in the candidate's bio	✓	×	×
Inferred	Gender	Probability of the candidate being male	✓	✓	✓

Table 5.1: Per-candidate features I extract from search results on **Indeed**, **Monster**, and **CareerBuilder**. I infer the last feature, *Gender*, based on each candidate's first name; other features are directly observed. Note that not all features are present on all hiring websites.

5000 candidates per query. The results display each candidate's name, current job title, and other features shown in the “I” column in Table 5.1. Clicking on a candidate opens their full resume, and there are no restrictions on how many resumes a recruiter may view. Figure 5.1a shows example search results from Indeed.

Monster shows 20 candidates per page in resume search results, up-to a maximum of 1000 candidates per query. The results page displays each candidate's name, current job title, and other features shown in the “M” column in Table 5.1. Figure 5.1b highlights the information that is shown for each candidate, and the pagination of results, on Monster. Clicking on a candidate opens their resume, but recruiters may only view 100 resumes per month. Unlike Indeed and CareerBuilder, Monster displays a *relevancy score* (between 0.0 and 10.0 with one decimal point of precision) next to each candidate in search results, which corresponds to the default sort ordering of the search results. It is unclear how the relevancy scores are calculated, and how Monster breaks ties between candidates with identical scores.

CareerBuilder's resume search engine shows 20–30 candidates per page, up-to a maximum of 5000 candidates per query. The information displayed per candidate is a small subset of what is shown on Indeed and Monster (see the “C” column in Table 5.1). CareerBuilder does not display relevancy scores, and recruiters may only view 50 resumes per month.

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

Category	Job Titles	U. S. State	Cities
First or Mid Level Officials & Managers	Marketing Manager, Business Development Manager, Casino Manager	Texas	Austin
Laborers	Laborer	Iowa	Des Moines
Office & Clerical Workers	Technical Recruiter, Payroll Specialist	Wisconsin	Madison, Milwaukee
Operatives	Truck Driver, Taxi Driver	Louisiana	New Orleans
Professionals	Human Resources Specialist, Physical Therapist, Occupational Therapist	New York	New York City, Buffalo
	Registered Nurse, Pharmacist, Speech Language Pathologist,	Nebraska	Omaha
	Software Engineer, Network Engineer, Mechanical Engineer,	Utah	Salt Lake City
	Electrical Engineer, Manufacturing Engineer, Accountant,	California	San Francisco, Stockton,
	Financial Analyst, Auditor, Tax Manager, Corrections Officer,		San Bernardino, Los Angeles
	Personal Trainer, Real Estate Agent	Missouri	Springfield
Sales Workers	Cashier, Retail Sales	Michigan	Detroit
Service Workers	Janitor, Bartender, Mail Carrier, Concierge, Call Center Director,	Ohio	Toledo, Cleveland, Cincinnati
	Customer Service and Support	Tennessee	Memphis
Technicians	Elevator Technician	Illinois	Chicago

Table 5.2: Job titles used in my queries, organized by EEOC category.

Table 5.3: Cities used in my queries.

5.2 Data Collection

In this section, I describe my data collection methodology, and the specific variables I extract from the data.

Crawl. To collect data for this study, I use an automated web browser to search for candidates on Indeed, Monster, and CareerBuilder. Intuitively, my crawler is designed to emulate how a recruiter would search for candidates on these hiring websites. I ran queries for 35 job titles in 20 American cities (described below) on all three sites, and recorded the resulting lists of candidates. On Indeed I recorded candidates' resumes, but not on Monster or CareerBuilder (since they only allow recruiters to view ≤ 100 resumes per month). All of my data was collected between May and October 2016, and the crawling took about 2 months on each site. My crawler sent queries every 30 seconds to minimize my impact on each site, and my data collection was IRB approved.

Search Parameters. To obtain a broad sample of candidates, I ran queries for 35 job titles in 20 cities. Table 5.2 lists my 35 job titles organized by Equal Employment Opportunity Commission [70] category. I chose these job titles because 19 of them are the most commonly searched job titles [47], while the remaining 16 do not require high-school-level education [41], which adds diversity to my query pool. Table 5.3 shows the 20 cities I focus on, which were chosen to give me broad geographic and demographic variety. I ultimately gathered 521,783, 265,172, and 67,580 candidates on Indeed, Monster, and CareerBuilder (respectively) after running the 35×20 queries on each site.

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

Candidate Features. Next, I extract information about candidates who appeared in the search results. I focus on three types of features: 1) profile data (e.g., experience, education level, activity, *etc.*); 2) inferred gender; and 3) the rank of each candidate in search results for a given query. Table 5.1 lists the features I am able to extract on each site. For example, Monster and CareerBuilder do not have *Information Relevance* or *Description Relevance*, while Indeed does not have *Skill Popularity*.

I normalize all features in Table 5.1 to be between 0 and 1 for consistency. Binary features, such as *Authorization*, *Relocate*, and *Skills Match*, are converted to 0 or 1. For other features, I apply the normalization procedures described below.

- *Job Title Relevance* and *Skills Relevance* refer to the normalized fraction of keywords in a candidate’s current job title and self-reported list of skills that match the terms in my query. For example, if I query for “software engineer” and a candidate’s current job title is “software designer”, I would assign them a *Job Title Relevance* of 0.5. I convert all words to their root form before computing the scores. Note that Monster allows candidates to enter three skills, hence I use three separate features on that site. In contrast, candidates on Indeed may enter as many skills as they wish, so I calculate the fraction of keyword matches on the aggregated skill set.
- *Education* is normalized using the ordered list of education-levels provided by Monster. The list contains “No Education”, “Some High School Coursework”, “High School or equivalent”, ..., “Bachelor’s Degree”, “Master’s Degree”, “Doctorate”, and “Professional”, from low to high. I normalize each candidate’s education levels uniformly: 0 being no education, and 1 being professional. Candidates on CareerBuilder must select from the same education-level list as those on Monster. Because Indeed allows free-text input of education-level (e.g., “ba”, “b.sc”, “be” all mean Bachelors degree), I parse the text and map it back to Monster’s education-level list.
- *Job Popularity* and *Skill Popularity* encode the popularity of each candidate’s current job title and skills. I normalize these features by applying min-max normalization on the popularity of the current job title or skill, where popularity is computed across all candidates in a given list of search results. For example, candidates with the most popular current job title would have *Job Popularity* of 1.0.
- *Last Modified* and *Experience* encode a candidate’s resume modification time and years of experience relative to all the other candidates in a given list of search results. For a given list,

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

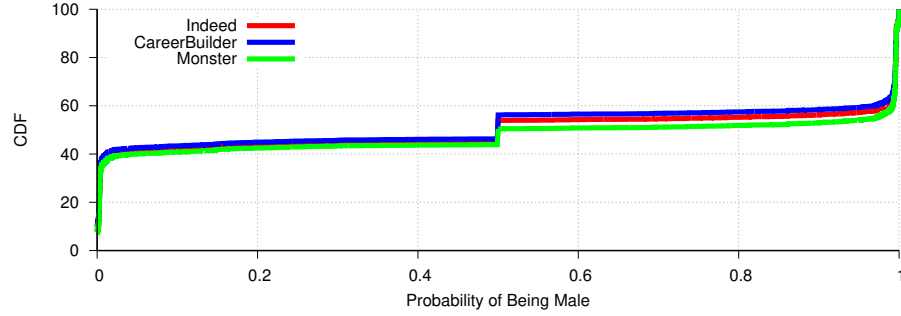


Figure 5.2: Inferred probability of being male for all candidates in my dataset.

I compute the Cumulative Density Function (CDF) of modification times/experience for all candidates. Each candidate’s feature value is then their relative rank in the CDF. For example, the candidate(s) with the most years of experience for a given query would have *Experience* of 1.0.

Inferring Gender. Since my goal is to examine resume search engines with respect to gender, I need to label each candidate’s gender. However, none of the sites I focus on collect this information. Instead, I infer each candidate’s gender based on their first name, which is a common method to infer gender in Western societies [65, 74, 105, 122, 178].

In this work, I rely on the U.S. baby name dataset [171] to infer candidates’ gender. Instead of treating gender as a binary variable, I assign a *probability of being male* to each candidate based on the fraction of times their first name was assigned to a male baby in the name dataset. I chose to represent gender as a probability since this corresponds to how a recruiter would perceive a candidate’s gender. For example, a candidate named “John” is almost certainly male, while a candidate named “Madison” is ambiguous.

Figure 5.2 shows the CDF of the probability of being male for all candidates on my dataset. I see that only 8% of candidates have ambiguous gender (scores around 0.5), meaning that I can be very confident in the vast majority of my gender labels. The plot also shows that the male/female ratio is approximately 1:1 on all three sites.

Ethics. I was careful to conduct my study in an ethical manner. This study was IRB approved. To protect the contextual privacy of candidates, I will not be releasing my crawled data. Furthermore, I used technical means to limit the impact of my crawlers on the target websites, and at no point did I contact candidates. In my controlled experiments (detailed in the next section), I only uploaded two resumes at a time to the hiring sites, meaning I decreased the rank of other candidates

by at most two. Although all three sites prohibit crawling in their terms of service, I believe my study is acceptable under the norms that have been established by prior algorithm audit studies that investigate civil rights issues [90, 162].

5.3 Analysis

In this section, I investigate the rankings of candidates produced by resume search engines with respect to gender. I organize my analysis around three questions: *first*, do the resume search engines exhibit *ranking bias*? *Second*, are the search engines *fair*, in the sense that candidates with equivalent attributes are assigned adjacent ranks? *Third*, if the resume search engines do exhibit ranking bias or unfairness with respect to gender, is this caused by the inclusion of gender as a feature, or via some other feature that is a proxy for gender? These questions are motivated by the realization that recruiters rely on resume search to actively recruit employees, the same way people rely on Google to surface relevant links. If a ranking algorithm is influenced by demographic variables, this might cause specific classes of candidates to consistently be ranked lower, thus harming their job prospects.

Throughout this section, I examine the top 1000 candidates returned in response to my queries. I do this for consistency, since Monster returns at most 1000 results per query, while Indeed and CareerBuilder may return more. The top 1000 candidates covers the first 20 pages of results returned by Indeed for a given query, all 50 pages from Monster, and the first 30-50 pages from CareerBuilder (the number of candidates per page varies).

Also recall that in this paper, I use the terms “top” and “high” to refer to desirable ranks in search results (e.g., rank 1), since this is the standard terminology used in IR literature [61, 104].

5.3.1 Ranking Bias

I begin by examining whether the search results returned by the three resume search engines exhibit *ranking bias*. **I define ranking bias as statistically significant differences in the distributions of ranks assigned to the members of two classes.** In my case, the two classes are male and female candidates. Ranking bias is analogous to *predictive bias* in the context of predictive algorithms [168], and it is a specific instance of the more general *classification bias* [107].

To identify ranking bias, I use the Mann-Whitney U (M-W U) test to compare the distribution of ranks for male and female candidates in a given list of search results [60]. The M-W U

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

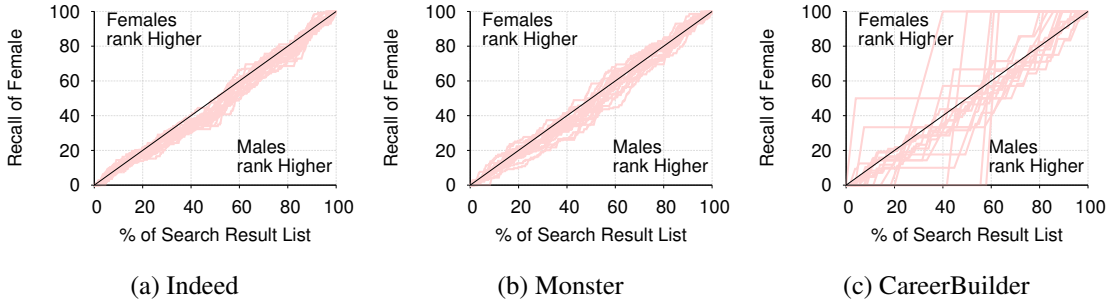


Figure 5.3: Recall of being female for candidates when searching for a “Software Engineer”. The 20 lines correspond to different cities. Lines below the diagonal mean that fewer female candidates appear than expected, given the overall population of candidates. Lines above the diagonal indicate the opposite, i.e., there are fewer male candidates than expected.

test is a nonparametric test of the null hypothesis that there is no tendency for ranks of one class to be significantly different from the other. I use the M-W U test because it works on unbalanced and discrete ranking data (unlike the Kolmogorov-Smirnov test). I omit search result lists where the number of female or male candidates is less than 20, as the result of M-W U is not reliable when samples are this small. Out of $35 \times 20 = 700$ samples on each hiring website, 648, 421, and 181 are suitable for analysis on Indeed, Monster, and CareerBuilder, respectively. I also remove candidates that have ambiguous gender ($Probability\ of\ Being\ Male \geq 0.2$ and ≤ 0.8); this eliminates 8% of candidates in my dataset.

Table 5.4 shows the results of the M-W U tests on the search result samples in my dataset. Each cell shows the percentage of samples across cities where the M-W U test returned $p \leq 0.05$ for a given job title and hiring website. For now, I focus on the columns showing test results computed on the original search results returned by the hiring websites.

I observe that there are several job titles the exhibit many instances of significant ranking bias. Most of these cases occur in blue-collar jobs (e.g., Truck Driver and Taxi Driver), engineering professions (e.g., Software, Network, and Electrical), and pink-collar jobs (e.g., Human Resources Specialist and Cashier). In other words, jobs that are historically gendered in the U. S. tend to exhibit ranking bias on the three hiring websites I survey. Overall, 18.2%, 13.5%, and 18.2% of job title/city pairs on Indeed, Monster, and CareerBuilder (respectively) show significant ranking bias.

To give a concrete example of ranking bias, I create recall plots such as Figure 5.3. To calculate recall, I iterate from the first candidate (i.e., rank 1) to the last candidate in a given search

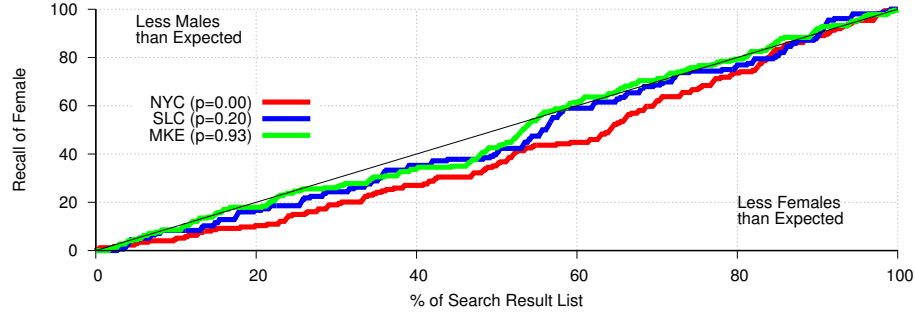


Figure 5.4: Recall curves for “Software Engineer” in three cities on Monster. All three curves show a discernable difference between the rank of female and male candidates, but this is not always detected by the Mann-Whitney U test.

result list R . At rank i , I calculate a tuple $(x_i, y_i) = (i/|R|, |R_{f,i}|/|R_f|)$, where $|R|$ is the total number of candidates in the list, and $|R_f|$ is the total number of *female* candidates in the list R . $|R_{f,i}|$ is the number of female candidates observed between ranks 1 and i . If female and male candidates are evenly distributed in the list, the resulting line is along the diagonal (marked with black); however, if females are consistently ranked lower then the line will be below the diagonal.

As an illustrative example, Figure 5.3 shows the recall for candidates on each hiring websites when I search for “Software Engineer”. Each red line corresponds to the search results in a different city. In this case, I see that most of the lines are below the diagonal, meaning that female candidates are under-represented in the search results relative to their overall percentage of the candidate population. In short: female candidates appear towards the bottom of the search results. The results in Figure 5.3c on CareerBuilder are very noisy; this occurs because CareerBuilder has the smallest candidate population of the three websites I survey, so for many job title/city pairs there are very few candidates. This leads to the abrupt changes in the plot lines.

On Indeed, 16 job titles out of 35 exhibit noticeable deviations from the diagonal, always in the favor of men. These jobs include Janitors and Laborers; Truck Drivers; Network, Electrical, and Software Engineers; Casino Manager; and Correction Officers. Similarly, on Monster, 12 job titles show trends in favor of men. These jobs include Truck Drivers; Network, Electrical, and Software Engineers; Laborers; and Corrections Officers. I observe similar trends on CareerBuilder for Software Engineers, Tax Managers, Pharmacists, Laborers, and Corrections Officers.

My recall plots highlight an important limitation of the M-W U test: it only rejects the null hypothesis when the recall line is significantly deviated from the diagonal line. Consider Figure 5.4,

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

which shows the recall curve and p value for “Software Engineer” in New York City (NYC), Salt Lake City (SLC), and Milwaukee (MKE) on Monster. I see that all three recall curves are similarly below the diagonal line (completely for NYC, partially for SLC and MKE), yet the p values vary dramatically. This demonstrates that there are even more search result lists in my dataset that exhibit ranking bias than Table 5.4 suggests.

Discussion. My findings demonstrate that the search results for many job titles consistently exhibit ranking bias across cities and across different hiring websites. In practice, this means is that one class of candidates (almost always females) appear lower in search results than the alternate class when recruiters search for these job titles.

However, just because the resume search engines that I have surveyed exhibit ranking bias, does not necessarily mean that the algorithms are *unfair*: it is possible that all candidates from class c_i have stronger features (e.g., more years of experience and/or more education) than candidates from class c_j , which causes the ranking algorithm to place the c_i candidates at higher positions. This issue occurs because ranking bias is an uncontrolled metric, meaning it does not take candidate features into account (beyond gender). I examine the fairness of the resume search engines in my study in the next section.

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

Job Title	Indeed			Monster			CareerBuilder		
	Original	Unbiased	Neutral	Original	Unbiased	Neutral	Original	Unbiased	Neutral
Accountant	25.0	0.0	25.0	20.0	0.0	1.3	12.5	0.0	12.5
Auditor	10.0	0.0	25.0	0.0	0.0	0.5	0.0	0.0	0.0
Bartender	20.0	0.0	20.0	10.0	0.0	1.0	16.7	0.0	16.7
Business Development Manager	0.0	0.0	0.0	5.6	0.0	11.1	0.0	0.0	0.0
Call Center Director	30.0	0.0	25.0	–	–	–	–	–	–
Cashier	15.0	0.0	15.0	10.0	0.0	0.8	42.1	0.0	47.4
Casino Manager	10.5	0.0	10.5	0.0	0.0	5.3	–	–	–
Concierge	5.6	0.0	11.1	9.1	0.0	0.5	0.0	0.0	0.0
Corrections Officer	25.0	0.0	20.0	0.0	0.0	2.1	–	–	–
Customer Service and Support	10.0	0.0	15.0	25.0	0.0	0.5	13.3	0.0	13.3
Electrical Engineer	27.8	0.0	27.8	25.0	0.0	7.0	–	–	–
Elevator Technician	0.0	0.0	0.0	–	–	–	–	–	–
Financial Analyst	11.1	0.0	11.1	11.1	0.0	11.1	41.7	0.0	25.0
Human Resources Specialist	15.0	0.0	30.0	0.0	0.0	5.3	75.0	0.0	75.0
Janitor	10.0	0.0	15.0	10.0	0.0	0.5	0.0	0.0	0.0
Laborer	25.0	0.0	25.0	45.0	0.0	1.3	0.0	0.0	0.0
Mail Carrier	15.0	0.0	15.0	14.3	0.0	14.3	–	–	–
Manufacturing Engineer	30.0	0.0	15.0	0.0	0.0	3.8	–	–	–
Marketing Manager	10.5	0.0	15.8	5.0	0.0	0.5	25.0	0.0	20.0
Mechanical Engineer	21.1	0.0	21.1	0.0	0.0	3.5	–	–	–
Network Engineer	52.6	0.0	47.4	0.0	0.0	6.6	0.0	0.0	0.0
Occupational Therapist	12.5	0.0	12.5	0.0	0.0	12.5	–	–	–
Payroll Specialist	10.0	0.0	20.0	0.0	0.0	14.3	0.0	0.0	0.0
Personal Trainer	10.5	0.0	10.5	0.0	0.0	14.3	0.0	0.0	0.0
Pharmacist	40.0	0.0	45.0	0.0	0.0	4.0	0.0	0.0	0.0
Physical Therapist	20.0	0.0	20.0	16.7	0.0	16.7	–	–	–
Real Estate Agent	10.0	0.0	0.0	5.3	0.0	5.3	0.0	0.0	0.0
Registered Nurse	10.0	0.0	10.0	5.9	0.0	11.8	0.0	0.0	0.0
Retail Sales	10.0	0.0	15.0	60.0	0.0	0.5	30.8	0.0	30.8
Speech Language Pathologist	33.3	0.0	33.3	–	–	–	–	–	–
Software Engineer	30.0	0.0	35.0	35.0	0.0	1.5	16.7	0.0	0.0
Tax Manager	10.0	0.0	10.0	0.0	0.0	1.1	0.0	0.0	0.0
Taxi Driver	33.3	0.0	33.3	0.0	0.0	33.3	–	–	–
Technical Recruiter	25.0	0.0	15.0	20.0	0.0	2.5	0.0	0.0	0.0
Truck Driver	20.0	0.0	20.0	14.3	0.0	14.3	50.0	0.0	0.0
Total Percentage	18.2	0.0	19.0	13.5	0.0	12.6	18.2	0.0	16.0

Table 5.4: Mann-Whitney U test on the rank of females and males in the top 1000 candidates per job title. Each cell shows the percentage (%) of cities with $p < 0.05$ for a given job title on a given hiring website. Cases where the percentage is $\geq 20\%$ are **bolded**. I show results for the original search results, as well as search results that have been re-ranked using my unbiased and neutral ranking algorithms. Note that I skip samples when the number of female or male candidates is less than 20; “–” marks instances where there are no cities with sufficiently large populations to test. I also remove the 8% of candidates with ambiguous genders.

5.3.2 Fairness

Next, I investigate whether the rankings of candidates produced by resume search engines are *fair* with respect to gender. **I define fairness as assigning adjacent ranks to candidates that have the same observable features (e.g., experience and education), but different genders.**

To investigate the fairness of the ranking algorithms, I use regression tests. My goal is to examine the effect of gender on candidate's rank while controlling for all other observable candidate features. If the gender feature is significant and has a non-zero coefficient in the fitted model, this indicates that the ranking algorithm in question is not fair, as candidates with equivalent features but different genders are not assigned the same rank. In this section, I will focus on the top 1000 and top 100 candidates in search results, since the former covers all candidates, while the latter focuses on the highly visible candidates that recruiters are likely to see [61, 158].

Model Specification. I adopt the Mixed Linear Model (MLM) for my regression tests. I regress on individual candidates, specifying the model as

$$y = \mathbf{X}\beta + \mu + \epsilon,$$

where y is a vector of the responses ($\log_2(\text{rank})$) of each candidate, explained below). \mathbf{X} is the design matrix, and the predictors include observed features and probability of being male for each candidate (see Table 5.1). β is the vector of fixed-effects parameters, including the global intercept. μ is the vector of random-effects intercepts. In other words, each group (*a.k.a.* search results for a specific job title and city) has a shared global fixed-effects intercept and an individual random-effects intercept.

Choosing a model that has fixed and random effects is important, because it agrees with two fundamental assumptions about my data:

1. Each hiring website has a single ranking algorithm for answering all queries, regardless of query term and location. This corresponds to fixed effects (i.e., feature weights) regardless of job title and location.
2. Candidates with identical features that appear in different search result lists may be assigned dramatically different ranks. Moreover, job titles and cities are a sampled subset of the entire population of candidates in the hiring websites' corpus. This corresponds to random-effects group intercepts that vary by job title and location.

The first assumption is reasonable because it is impractical for a hiring website to implement a different ranking algorithm for an unbounded number of possible search terms. The second

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

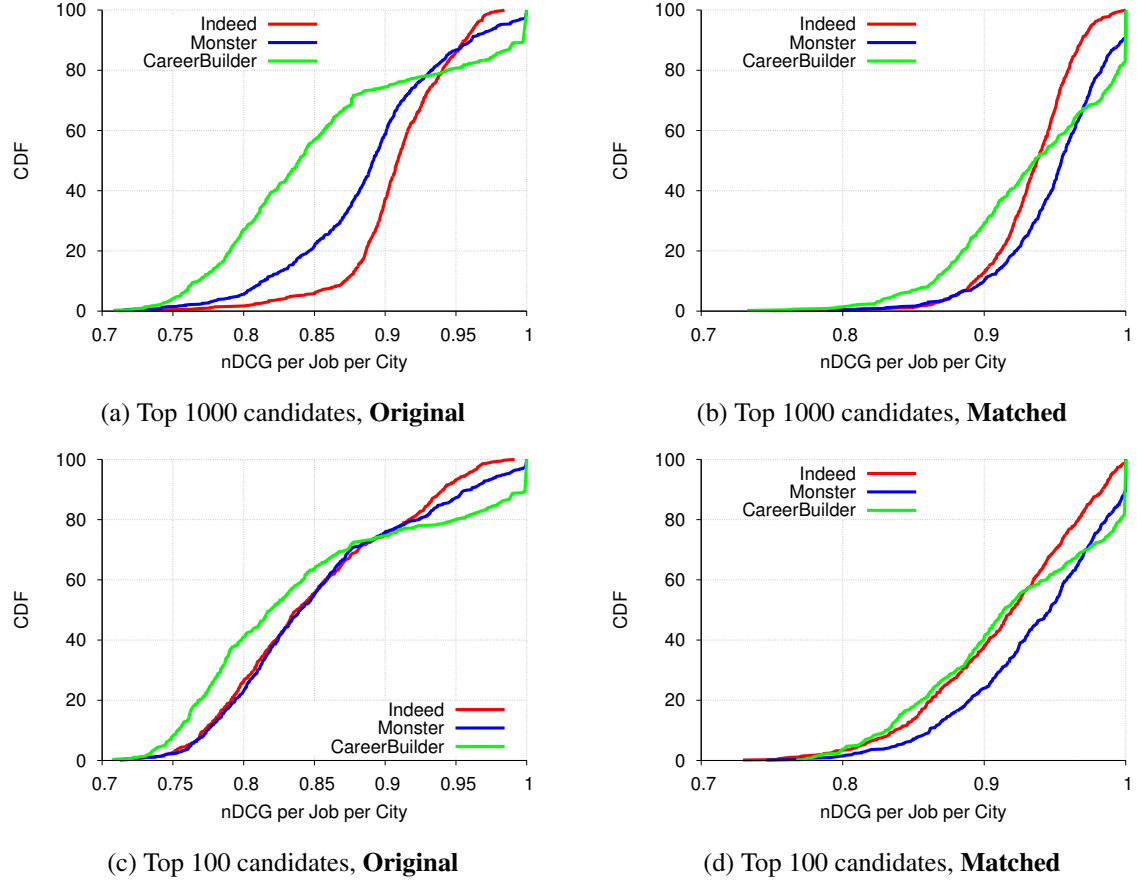


Figure 5.5: nDCG comparison of the predicted rankings $\hat{\mathbf{R}}$ produced by the mixed linear models versus the original rankings \mathbf{R} . These results were computed using the **Original** and **Matched** candidates from my dataset.

assumption holds because the population of candidates, and their relative qualifications, varies across locations and professions. For example, consider two male Software Engineer candidates with identical qualifications that live in Mobile, AL and San Francisco, CA. The former candidate may appear at a high rank in the Mobile search results, while the latter appears at a low rank San Francisco, even though they have the same features, because the latter location has a larger population of highly qualified candidates in this specific profession. In other words, equivalent \mathbf{X} does not mean equivalent \mathbf{y} if the job title and/or location are different. The MLM uses the same fixed effects vector β across all samples (corresponding to assumption 1), but different random effects intercepts μ per sample (which permits variance across job titles and cities, corresponding to assumption 2).

For the dependent variable in my regressions, I use $\log_2(\text{rank})$. I use this transformation

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

Feature	Dependent Variable: $\log_2(\text{rank})$											
	Indeed				Monster				CareerBuilder			
	Top 1000		Top 100		Top 1000		Top 100		Top 1000		Top 100	
	Original	Matched	Original	Matched	Original	Matched	Original	Matched	Original	Matched	Original	Matched
Fixed Effect Intercept	7.125***	7.281***	4.803***	4.826***	6.991***	7.461***	5.938***	6.275***	4.47***	5.528***	4.129***	4.601***
Job Title Relevance	-1.196***	-0.783***	-0.518***	-0.295***	-1.628***	-1.449***	-1.3***	-1.199***	-1.57***	-1.522***	-1.258***	-1.204***
Skills Relevance (1)	-0.14***	-0.275***	-0.051	-0.006	-0.268***	-0.065**	-0.31***	-0.046				
Skills Relevance (2)					-0.143***	0.002	-0.109***	0.17*				
Skills Relevance (3)					-0.096***	0.053	-0.108***	0.151				
Education level	0.086***	0.117***	0.042**	0.106***	-0.079***	-0.011	-0.061*	-0.009	-0.038*	-0.074**	-0.027	-0.034
Job Popularity	-0.084***	-0.098***	-0.115***	-0.184***	-0.163***	-0.177***	-0.004	0.009	-0.193***	-0.242***	-0.147***	-0.175***
Last Modified	-2.02***	-1.631***	-2.053***	-1.738***	-0.203***	-0.213***	-0.197***	-0.202***	-0.139***	-0.109***	-0.149***	-0.181***
Experience	0.217***	0.279***	0.116***	0.065*	-1.041***	-0.853***	-1.303***	-1.359***	-0.106***	-0.104***	-0.185***	-0.226***
Relocate					-0.019***	-0.015	-0.021	-0.048				
Skills Popularity (1)					-0.08***	-0.099***	-0.048**	-0.093**				
Skills Popularity (2)					-0.086***	-0.103***	-0.062**	-0.103*				
Skills Popularity (3)					-0.103***	-0.113***	-0.017	-0.018				
Bio Relevance	0.046***	0.073**	0.041	0.135								
Information Relevance	-0.32***	-0.251***	-0.255***	-0.134								
Skills Match	-0.034	0.066	-0.072	-0.313								
Information Match	-0.042**	0.037	-0.093*	-0.034								
Bio Match	-0.189	-0.086***	-0.262***	-0.36***								
Random Effect (s.d.)	0.072	0.062	0.01	0.008	0.27	0.231	0.106	0.066	0.229	0.196	0.018	0.118
Prob. of Being Male	-0.019***	-0.012*	-0.042***	-0.051**	-0.043***	-0.025**	-0.028*	-0.051*	-0.039**	-0.02	-0.071***	-0.055*
Observations	521783	179630	67410	18630	265172	83862	50813	11836	67580	30502	28289	13205

Table 5.5: Estimated coefficients and standard deviation of mixed linear regressions on the top 1000 and top 100 **Original** and **Matched** candidates in search results from each hiring website, grouped by city and job title. Significance level is unavailable for *Random Effect*. Note: * $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$.

for two reasons:

- The $\log()$ function is monotonically increasing, which maintains the ordering of candidates after the transformation.
- It prioritizes top-ranked candidates by giving them higher weight, while decaying the importance for lower-ranked candidates.

Logarithms have been found to be widely applicable in the IR literature. Empirical studies [61, 158], backed up by eye-tracking surveys [12, 78, 82], have found that the probability of search result items being clicked decays logarithmically. Similarly, $\log_2()$ is used to calculate normalized Discount Cumulative Gain (nDCG), which is a standard metric used to quantify the similarity of two search result lists [102, 103]. By using logarithmic decay, nDCG affords higher weight to the top items in the search result lists, since they are known to be of higher practical importance.

Matching. I analyze two populations of candidates in my regressions: the **Original**

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

Top k Candidates	Indeed Gender Coef.	Monster Gender Coef.	CareerBuilder Gender Coef.
Top 2	-0.080**	-0.032	0.008
Top 5	-0.072*	-0.062*	0.006
Top 10	-0.023	-0.031	0.027
Top 20	-0.009	-0.056*	-0.023
Top 50	-0.036*	-0.047**	-0.053*
Top 100	-0.042***	-0.028*	-0.071***
Top 200	-0.029**	-0.026**	-0.071***
Top 500	-0.022***	-0.040***	-0.055***
Top 1000	-0.019***	-0.043***	-0.039**

Table 5.6: Estimated *Probability of Being Male* coefficient from mixed linear regressions as the length of the search result list is varied. These results were computed using the **Original** candidates from my dataset. *Note:* * $p < 0.05$; ** $p < 0.01$; *** $p < 0.001$.

population, which includes all candidates, and a **Matched** subset of candidates. The goal of matching is to reduce selection bias in a dataset by creating a subsample that 1) includes a balanced population (in my case, between male and female candidates) and 2) is representative of the overall dataset. This is done by finding pairs of male and female candidates whose features are as similar as possible, but differ along the examined variable (rank) [95].

To construct my matched subpopulation, I leverage the MatchIt software developed by Ho et al. [96]. MatchIt implements several nonparametric matching methods (e.g., *exact matching*, *coarsened exact matching (CEM)*, *nearest neighbor*, etc.) that make no assumptions about the relationship between rank and visible features, which gives me a way to control for unobserved features, beyond what is visible in the search result data.

In this study I use the CEM method, which is a relaxed version of *exact matching*: the feature values are binned, creating more flexibility to find matches.² For categorical features (e.g., *Searched Job Title*, *Searched City*, *Education*, etc.), I set each category as a separate bin and use exact matching. For *Experience* I set seven bins, specified in years of experience: 0–1, 1–5, 5–8, 8–10, 10–15, 15–20, and 20+. For *Last Modified* I set five bins: 1–7 days, 7–30 days, 30–90 days, 90–365 days, 1+ years.³ I chose to manually specify bins for these two continuous features so that each

²I tried to use *exact matching*, but it could only produce subsamples containing <1% of candidates. The fundamental problem is that the feature-space in my dataset is large (I have many features, each of which has many possible values), which makes it very difficult to find exactly matching pairs of candidates.

³I use seven and five bins for *Experience* and *Last Modified* when examining the top 1000 candidates. I reduce the number of bins to four and three, respectively, when examining the top 100 candidates, to compensate for the reduced

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

bin would correspond to a human-interpretable timerange. For the remaining continuous features, I leverage the default binning algorithm in MatchIt. Tables 5.7 and 5.8 show the features I match on each hiring website,⁴ as well as each feature’s type, the number of bins, and whether MatchIt’s default binning was used.

Validation. Before regressing on the matched subpopulations, I must first verify that they are large and representative. Tables 5.7 and 5.8 show the descriptive statistics of all candidates in the top 1000 and top 100 populations, versus the corresponding matched subpopulations produced by CEM for Indeed, Monster, and CareerBuilder, respectively. I make two key observations: *first*, CEM identifies tens of thousands of matching pairs on each website (unlike *exact* matching), which means the populations are large enough to analyze. *Second*, when I examine the difference in means for the control features in the matched pairs, I observe that the differences are all close to zero, which demonstrates that the matched pairs have extremely similar features. Furthermore, the improvement to the mean difference is almost always $>90\%$, which shows that the matched subpopulation is substantially more balanced than the overall population. Taken together, these observations demonstrate that my matched subpopulations are suitable for analysis.

Model Fitting. For each hiring website I fit four MLMs: two on the top 1000 candidates in each search result list, and two on the top 100 candidates. For each top k , I fit models using my **Original** candidates and the **Matched** subset of candidates. Negative coefficients in my model signify effects with higher rank.⁵

I conduct the Variance Inflation Factors (VIF) test to remove multicollinearity before fitting the models; all the variables remain after the test. I also give the correlation matrix in Table 5.9 for the reader’s reference.

To assess how well the MLMs fit for my data, I evaluate their predictive power. To do this, I treat the each model as a ranking algorithm: I input the ground-truth feature vectors of candidates from a given search result list \mathbf{R} into a fit model, which then outputs a predicted log ranking $\hat{\mathbf{y}}$ (corresponding to a predicted ranking $\hat{\mathbf{R}}$). Next, I use the nDCG metric to compute the similarity between $\hat{\mathbf{R}}$ and the original ranking \mathbf{R} . nDCG is a standard metric used in the IR literature to

range of values for these features that I observe in the more restricted population.

⁴I do not include the *Job Popularity* and *Skill Popularity* features, as these are not directly observed from the candidate information, but calculated by me using data from all candidates.

⁵Note that a smaller $\log_2(\text{rank})$ value translates to a higher and more desirable rank for a candidate, because $\log_2(\text{rank})$ increases with rank.

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

compare ranked lists [102, 103]. The DCG of a ranking $R = [r_1, r_2, \dots, r_k]$ is calculated as

$$\text{DCG}(\mathbf{R}) = g(r_1) + \sum_{i=2}^k (g(r_i) / \log_2(i))$$

where $g(r_i)$ is the “gain” or score assigned to result r_i . nDCG is defined as $\text{DCG}(\hat{\mathbf{R}}) / \text{DCG}(\mathbf{R})$, where \mathbf{R} is the ideal ranking of the items in $\hat{\mathbf{R}}$. In my case, \mathbf{R} is the original ranking produced by a hiring site (i.e., I treat the original ranking as the baseline), and $\hat{\mathbf{R}}$ is a ranking predicted by the model. An nDCG value of 1 indicates that the two rankings are identical. In essence, nDCG for rank responses is analogous to R^2 for continuous responses and misclassification error for classification problems.

Figure 5.5 shows the evaluation of the predictive power of the MLMs fit to my **Original** and **Matched** candidates. I observe that 99%, 95%, and 73% of the nDCG scores are ≥ 0.8 for top 1000 candidates when fit to the **Original** data on Indeed, Monster, and CareerBuilder, respectively. For the **Matched** subsamples of top 1000 candidates, 99%, 99%, and 93% of the nDCG scores are ≥ 0.85 on these three websites. The nDCG scores are slightly lower for top 100 candidates: 60% – 77% of the nDCG scores are ≥ 0.8 across all three websites for the **Original** data, while 83% – 93% of the nDCG scores are ≥ 0.85 across all three websites for **Matched** candidates. These results demonstrates that my MLMs can reproduce most of the search results regardless of the underlying population of candidates (top 1000 or 100, **Original** or **Matched**), which indicates that the models are a good fit for my data.

Results. Table 5.5 shows the results of my twelve MLM regressions. I observe that the majority of features have significant, negative effect on log rank on all three hiring sites, such as *Job Title Relevance*, *Job Popularity*, and *Last Modified*. On Indeed, *Last Modified* has the largest coefficient by far, which matches their documentation stating that they tend to rank candidates by the time of their most recent resume update (from most to least recent) [100]. Interestingly, *Education Level* and *Experience* have significant, positive effects on log rank on Indeed, which suggest that there are more candidates with lower degrees and less experience in the top ranks (possibly newly graduated students). In contrast, Monster and CareerBuilder both exhibit significant, negative effects of *Experience* on log rank.

Overall, the sign of coefficients and their significance show remarkable consistency across the models fit to different populations. There are only a few features that have a significant, negative impact on log rank in the top 1000 populations, but not the top 100. Examples include *Skill Relevance (1)* on Indeed, *Job Popularity* on Monster, and *Education* on CareerBuilder. Less common are cases

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

where the **Original** and **Matched** populations disagree, such as *Information Relevance* on Indeed, and *Education* on Monster.

Most importantly, I observe that the *Probability of Being Male* feature is significant and negative in 11 of the 12 models. This indicates that men rank higher than women with equivalent features, thus demonstrating that the ranking algorithms used by these three hiring websites are unfair (according to my definition of fairness).

To further bolster my claims, I fit MLM regressions to the top k **Original** candidates on each hiring website. I show the resulting *Probability of Being Male* coefficients and p -values in Table 5.6. I see that the gender coefficient on each site is consistently significant and negative for $k \geq 50$. Furthermore, the coefficients are negative (and sometimes significant) beginning at $k = 2$ on Indeed and Monster. These results demonstrate that the gender effects are robust to changes in k beyond a low minimum floor, i.e., unfairness can be detected even in relatively small subsets of the top candidates.

Discussion. At this point in my analysis, it is unclear why the ranking algorithms exhibit unfairness. One possibility is that the hiring websites infer candidate's gender (since they do not directly ask candidates to disclose this information), and then use gender as a feature in their algorithms. However, given the sensitivity surrounding issues of gender discrimination in hiring, this possibility seems unlikely. Another possibility is that the ranking algorithms incorporate some feature (or features) that is correlated with gender, but that I cannot observe in my search result dataset (and thus cannot control). I delve into these questions in the next section.

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

Category	Male	Female	Feature	Binning	All Data			Matched Data			Improvement Mean Diff
					Means Male	Means Female	Mean Diff	Means Male	Means Female	Mean Diff	
			distance	–	0.5176	0.4877	0.0299	0.4924	0.4925	-0.0002	99.4323
			Job Title	cat, 35, A	17.4159	14.3647	3.0512	16.0758	16.1012	-0.0254	99.1674
			City	cat, 20, A	9.6171	9.4842	0.1329	9.2912	9.2934	-0.0021	98.3945
			Job Title Relevance	num, 7, A	0.3004	0.2727	0.0277	0.2078	0.2078	0	100
			Skills Relevance (1)	num, 7, A	0.0496	0.0415	0.0081	0.0043	0.0043	0	100
			Education	cat, 10, A	0.4679	0.4627	0.0053	0.4301	0.4301	0	100
			Last Modified	num, 5, M	0.5073	0.5019	0.0054	0.5004	0.5007	-0.0003	94.4605
			Experience	num, 7, M	0.5173	0.5001	0.0172	0.4999	0.4993	0.0006	96.5701
			Bio Relevance	num, 7, A	0.135	0.114	0.0211	0.0244	0.0244	0	100
			Information Relevance	num, 7, A	0.3579	0.3501	0.0077	0.2444	0.2444	0	100
			Skills Match	cat, 2, A	0.0086	0.0075	0.0011	0.0004	0.0004	0	100
			Information Match	cat, 2, A	0.0239	0.0219	0.002	0.002	0.002	0	100
			Bio Match	cat, 2, A	0.0741	0.0919	-0.0178	0.0445	0.0445	0	100
			Rank	–	457.881	454.6293	3.2516	484.0105	480.9978	3.0128	7.346

(a) Sample Size for Indeed,
Top 1000

(b) Balance Checking Statistics for Indeed, Top 1000

Category	Male	Female	Feature	Binning	All Data			Matched Data			Improvement Mean Diff
					Means Male	Means Female	Mean Diff	Means Male	Means Female	Mean Diff	
			distance	–	0.5408	0.5206	0.0201	0.5522	0.5521	0	99.8889
			Job Title	cat, 35, A	16.8625	14.6349	2.2276	17.9649	18.0011	-0.0361	98.3775
			City	cat, 20, A	9.8693	9.7387	0.1306	9.8414	9.838	0.0034	97.3949
			Job Title Relevance	num, 7, A	0.3377	0.3693	-0.0315	0.2174	0.2174	0	100
			Skills Relevance (1)	num, 7, A	0.1241	0.1492	-0.0251	0.0566	0.0566	0	100
			Skills Relevance (2)	num, 7, A	0.0847	0.0923	-0.0076	0.0213	0.0213	0	100
			Skills Relevance (3)	num, 7, A	0.0693	0.0747	-0.0054	0.0148	0.0148	0	100
			Education	cat, 12, A	0.5876	0.5811	0.0064	0.585	0.585	0	100
			Last Modified	num, 5, M	0.5103	0.517	-0.0067	0.5236	0.5242	-0.0006	90.7378
			Experience	num, 7, M	0.5248	0.4923	0.0326	0.507	0.505	0.0021	93.6805
			Relocate	cat, 2, A	0.6541	0.6109	0.0431	0.7054	0.7054	0	100
			Rank	–	384.7604	410.1525	-25.3921	448.9949	454.3637	-5.3688	78.8566

(c) Sample Size for Monster,
Top 1000

(d) Balance Checking Statistics for Monster, Top 1000

Category	Male	Female	Feature	Binning	All Data			Matched Data			Improvement Mean Diff
					Means Male	Means Female	Mean Diff	Means Male	Means Female	Mean Diff	
			distance	–	0.4977	0.4759	0.0218	0.4908	0.4909	-0.0001	99.6468
			Job Title	cat, 35, A	15.6974	13.7541	1.9434	14.6425	14.6774	-0.0349	98.2032
			City	cat, 20, A	9.4957	9.1079	0.3878	9.3551	9.3538	0.0013	99.6755
			Job Title Relevance	num, 7, A	0.3798	0.3866	-0.0068	0.3259	0.3259	0	100
			Education	num, 7, A	0.4632	0.4039	0.0593	0.4672	0.4672	0	100
			Last Modified	num, 5, M	0.5004	0.503	-0.0026	0.4958	0.4971	-0.0013	51.2495
			Experience	num, 7, M	0.5416	0.5165	0.0251	0.5365	0.5351	0.0014	94.3952
			Rank	–	209.7027	247.7418	-38.0391	290.333	291.0539	-0.7208	98.105

(e) Sample Size for Career-
Builder, Top 1000

(f) Balance Checking Statistics for CareerBuilder, Top 1000

Table 5.7: Coarsened Exact Matching statistics for all three hiring websites on the top 1000 candidates. Note that *Rank* is not a feature in the matching procedure; I list it here to check the balance statistics. The “Binning” column shows the type of each feature (categorical or numerical), how many bins were used, and whether those bins were chosen **Automatically** by MatchIt or **Manually** by me.

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

Category	Male	Female	Feature	Binning	All Data			Matched Data			Improvement Mean Diff
					Means	Means	Mean	Means	Means	Mean	
					Male	Female	Diff	Male	Female	Diff	
			distance	–	0.5126	0.4903	0.0223	0.4941	0.4951	-0.001	95.5994
			Job Title	cat, 35, A	17.9644	15.4478	2.5166	16.7918	16.9282	-0.1364	94.5795
			City	cat, 20, A	9.5852	9.431	0.1542	9.6072	9.5984	0.0089	94.2436
			Job Title Relevance	num, 7, A	0.4091	0.3923	0.0168	0.3211	0.3211	0	100
			Skills Relevance (1)	num, 7, A	0.0544	0.0475	0.0068	0.0054	0.0054	0	100
			Education	cat, 10, A	0.466	0.4784	-0.0123	0.4182	0.4182	0	100
			Last Modified	num, 3, M	0.3307	0.3345	-0.0038	0.29	0.2874	0.0027	30.6878
			Experience	num, 4, M	0.482	0.4683	0.0137	0.4331	0.4334	-0.0003	97.7444
			Bio Relevance	num, 7, A	0.1333	0.1118	0.0215	0.02	0.02	0	100
			Information Relevance	num, 7, A	0.3863	0.3897	-0.0034	0.2594	0.2594	0	100
			Skills Match	cat, 2, A	0.0109	0.0086	0.0023	0.0009	0.0009	0	100
			Information Match	cat, 2, A	0.0269	0.0227	0.0042	0.0023	0.0023	0	100
			Bio Match	cat, 2, A	0.0932	0.1161	-0.0229	0.0538	0.0538	0	100
			Rank	–	49.7075	50.2131	-0.5057	49.2912	50.6214	-1.3302	-163.0531

(a) Sample Size for Indeed, Top 100

(b) Balance Checking Statistics for Indeed, Top 100

Category	Male	Female	Feature	Binning	All Data			Matched Data			Improvement Mean Diff
					Means	Means	Mean	Means	Means	Mean	
					Male	Female	Diff	Male	Female	Diff	
			distance	–	0.5714	0.5518	0.0196	0.5667	0.5668	-0.0002	99.224
			Job Title	cat, 35, A	17.679	15.149	2.53	17.38	17.4438	-0.0638	97.4788
			City	cat, 20, A	9.7571	9.6203	0.1368	9.7278	9.7089	0.0189	86.1649
			Job Title Relevance	num, 7, A	0.5593	0.577	-0.0177	0.5244	0.5244	0	100
			Skills Relevance (1)	num, 7, A	0.1743	0.1813	-0.007	0.0963	0.0963	0	100
			Skills Relevance (2)	num, 7, A	0.1129	0.1083	0.0045	0.0319	0.0319	0	100
			Skills Relevance (3)	num, 7, A	0.0869	0.082	0.0049	0.0147	0.0147	0	100
			Education	cat, 12, A	0.6069	0.6091	-0.0022	0.6184	0.6184	0	100
			Last Modified	num, 3, M	0.5086	0.5198	-0.0112	0.5302	0.5326	-0.0024	78.8243
			Experience	num, 4, M	0.5735	0.5627	0.0108	0.5829	0.5799	0.003	71.8669
			Relocate	cat, 2, A	0.6545	0.6127	0.0418	0.716	0.716	0	100
			Rank	–	46.1663	46.7084	-0.5421	48.0163	49.7964	-1.78	-228.3505

(c) Sample Size for Monster, Top 100

(d) Balance Checking Statistics for Monster, Top 100

Category	Male	Female	Feature	Binning	All Data			Matched Data			Improvement Mean Diff
					Means	Means	Mean	Means	Means	Mean	
					Male	Female	Diff	Male	Female	Diff	
			distance	–	0.5394	0.5149	0.0245	0.5333	0.5336	-0.0002	99.0317
			Job Title	cat, 35, A	16.7937	14.0781	2.7156	16.0784	16.1792	-0.1008	96.2887
			City	cat, 20, A	9.3992	9.118	0.2812	9.3211	9.35	-0.0289	89.7113
			Job Title Relevance	num, 7, A	0.4818	0.4967	-0.0148	0.4808	0.4808	0	100
			Education	num, 7, A	0.459	0.4246	0.0344	0.4589	0.4589	0	100
			Last Modified	num, 3, M	0.4969	0.4988	-0.0019	0.4922	0.4881	0.0041	-118.9678
			Experience	num, 4, M	0.5434	0.5143	0.0291	0.534	0.5271	0.0069	76.3071
			Rank	–	38.6497	40.4052	-1.7555	41.7055	42.1318	-0.4263	75.7162

(e) Sample Size for CareerBuilder, Top 100

(f) Balance Checking Statistics for CareerBuilder, Top 100

Table 5.8: Coarsened Exact Matching statistics for all three hiring websites on the top 100 candidates. Note that *Rank* is not a feature in the matching procedure; I list it here to check the balance statistics. The “Binning” column shows the type of each feature (categorical or numerical), how many bins were used, and whether those bins were chosen **Automatically** by MatchIt or **Manually** by me.

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

Variable	Mean	s.d	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)
(1) Job Title Relevance	0.29	0.38											
(2) Skills Relevance (1)	0.05	0.18	0.02										
(3) Education level	0.47	0.35	0.11	0.04									
(4) Job Popularity	0.04	0.19	0.33	0.00	-0.02								
(5) Last Modified	0.50	0.29	0.10	-0.04	-0.03	0.05							
(6) Experience	0.50	0.29	0.04	-0.02	0.00	-0.05	0.18						
(7) Bio Relevance	0.13	0.27	0.09	0.40	0.07	0.02	0.02	0.01					
(8) Information Relevance	0.35	0.38	0.11	0.08	0.08	0.00	0.05	0.07	0.11				
(9) Skills Match	0.01	0.09	-0.01	0.44	-0.03	0.01	-0.01	-0.01	0.17	-0.01			
(10) Information Match	0.02	0.15	0.04	0.17	-0.01	0.04	0.01	0.00	0.43	-0.01	0.42		
(11) Bio Match	0.08	0.28	0.03	-0.01	-0.04	0.03	0.04	0.02	-0.03	0.45	0.05	0.06	
(12) Prob. of Being Male	0.50	0.46	0.03	0.02	0.01	-0.01	0.01	0.03	0.04	0.01	0.01	0.01	-0.03

(a) Indeed

Variable	Mean	s.d	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
(1) Job Title Relevance	0.36	0.43												
(2) Skills Relevance (1)	0.14	0.29	0.26											
(3) Skills Relevance (2)	0.09	0.24	0.18	0.08										
(4) Skills Relevance (3)	0.07	0.22	0.15	0.08	0.10									
(5) Education level	0.59	0.24	0.05	0.09	0.08	0.08								
(6) Job Popularity	0.12	0.30	0.38	0.13	0.06	0.05	-0.05							
(7) Last Modified	0.51	0.29	0.02	0.00	0.00	0.00	0.01	0.02						
(8) Experience	0.50	0.29	-0.09	0.00	0.00	-0.01	0.02	-0.07	0.05					
(9) Relocate	0.64	0.48	0.00	-0.01	-0.01	-0.01	0.04	0.00	0.09	-0.07				
(10) Skills Popularity (1)	0.31	0.38	0.22	0.58	0.02	0.02	0.03	0.15	0.00	0.00	-0.01			
(11) Skills Popularity (2)	0.19	0.30	0.15	0.01	0.53	0.03	0.02	0.08	0.01	0.00	0.00	0.02		
(12) Skills Popularity (3)	0.14	0.26	0.10	-0.01	0.03	0.50	0.01	0.06	0.00	-0.01	0.00	-0.02	0.02	
(13) Prob. of Being Male	0.53	0.47	-0.04	-0.04	-0.01	-0.01	0.01	-0.03	-0.01	0.06	0.04	-0.06	-0.03	-0.02

(b) Monster

Variable	Mean	s.d	(1)	(2)	(3)	(4)	(5)
(1) Job Title Relevance	0.39	0.43					
(2) Education level	0.44	0.36	0.03				
(3) Job Popularity	0.13	0.30	0.50	-0.02			
(4) Last Modified	0.50	0.29	0.00	0.03	0.00		
(5) Experience	0.52	0.28	-0.13	-0.01	-0.10	0.04	
(6) Prob. of Being Male	0.49	0.46	-0.01	0.08	-0.04	0.00	0.04

(c) CareerBuilder

Table 5.9: Means, standard deviation, and correlations of the variables on Indeed, Monster, and CareerBuilder. These results were computed using the **Original** candidates from my dataset.

5.3.3 Uncovering Hidden Features

Up to now, my analysis has focused on candidate features that are directly observable in the search results. However, there is an element of each candidates' profile that I cannot observe (on Monster and CareerBuilder), but that may be taken into account by the ranking algorithm: resumes. For example, Monster and CareerBuilder ask candidates to enter their education level into their profile, and display this information in search results (e.g., high school diploma or Ph.D.), but actual almae matres are likely stated in each candidate's resume. The ranking algorithm could parse this additional information from the PDF-format resume and use it when ranking candidates. Parsing resumes makes sense from a website design standpoint, in that it allows the websites to collect detailed information about candidates without having to ask them to enter it manually, which can be tedious and time consuming.

To test if resume content influences ranking, I conducted controlled experiments using resumes created and uploaded by me. I create two user accounts, *A* and *B*, in that temporal order, with identical profile information and resumes. I then verify that the two accounts appear directly adjacent in search results in the order *B*, *A*. Next, I delete the old resumes, upload two new resumes (starting with *A*) that differ by exactly one feature, then query for my users again.⁶ In each treatment, I assign *A* the "stronger" value for the feature, e.g., *A* attended an Ivy League school while *B* attended community college. If user *A* appears *before* user *B*, it means the treatment variable in the resumes has flipped the rank ordering, thus revealing that the algorithm takes that particular resume feature into account. I repeat this process eight times on all three hiring websites, with the different treatment features shown in Table 5.10.

Table 5.10 shows the results of my controlled experiments. Crucially, my gender treatment did not influence the order of my users, which confirms that none of the hiring websites explicitly rely on gender as a feature. In fact, of my eight treatments, I see that only three features were picked out by the ranking algorithms: Monster's algorithm ranks users by the strength of their alma mater and whether they are currently employed, while CareerBuilder's algorithm takes employment status and frequency of job changes into account.

Discussion. While the use of these three features is not particularly surprising, this

⁶The time delay between uploading a resume and having it appear in search results can vary between 4–6 hours; during this time I query periodically until my users appear. Thus, it is unlikely that my users receive many (if any) clicks from recruiters before I measure their ranks. Minimizing clicks is important, because clicks may be a feature used to rank candidates.

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

Feature	Description	On Indeed	On Monster	On CareerBuilder
Gender	Candidate <i>A</i> is male and candidate <i>B</i> is female	No Effect	No Effect	No Effect
School ranking	Candidate <i>A</i> lists Harvard and candidate <i>B</i> lists a less known university	No Effect	$A.rank > B.rank$	No Effect
Employment	Candidate <i>A</i> is currently employed while candidate <i>B</i> is not	No Effect	$A.rank > B.rank$	$A.rank > B.rank$
Number of keywords	Candidate <i>A</i> has more keywords matching the query than candidate <i>B</i>	No Effect	No Effect	No Effect
Resume length	Candidate <i>A</i> has a longer resume than candidate <i>B</i>	No Effect	No Effect	No Effect
Job Churn	Candidate <i>A</i> changes jobs less often than candidate <i>B</i>	No Effect	No Effect	$A.rank > B.rank$
Contact information	Candidate <i>A</i> has more detailed contact information than candidate <i>B</i> , e.g., social network accounts	No Effect	No Effect	No Effect
Company name	Candidate <i>A</i> is currently employed at Google and candidate <i>B</i> is in a made-up company	No Effect	No Effect	No Effect

Table 5.10: Features tested in my controlled resume experiments. Cases where candidate *A* ranks higher than *B* reveals that the given feature is taken into account by the ranking algorithm.

offers a possible explanation for why these ranking algorithms exhibit unfairness. Although I control for all visible features in my matching and regression tests, I cannot control these unobserved features. If these unobserved features are correlated with gender, then their influence will manifest in the MLM in the coefficient of the gender feature.

It is plausible that these hidden features may correlate with gender. For example, a woman returning to the workforce after a lapse in employment to raise children will be ranked lower by Monster and CareerBuilder’s algorithms than an equivalently experienced, but currently employed, man. In other words, structural inequalities that are related to gender can be exposed to the ranking algorithms through candidates’ resumes.

Lastly, it is important to note that the treatments I examine in Table 5.10 may not be comprehensive: I do not know what other hidden features the hiring websites extract from resumes. I manually examined many real resumes from the hiring websites to inform the construction of my tests, but I make no claim to have covered all potential features.

5.4 Alternative Ranking Approaches

In the previous section, I show that there are two pervasive issues with the search results generated by Indeed, Monster, and CareerBuilder: the results exhibit ranking bias, and they are unfair, with respect to gender. This raises critical questions: *can these problems be remedied?* If so, *are their tradeoffs when attempting to address bias or unfairness?*

In this section, I propose two alternative ranking methods that each obey a different fairness constraint: an *unbiased* ranker and a *neutral* ranker. I describe how I implement these rankers, and then compare their output to the original search results produced by the hiring websites, as well as to each other.

5.4.1 Algorithms

I begin by introducing my two proposed ranking algorithms and the types of fairness that they guarantee.

Unbiased Ranker. The goal of my unbiased ranker is to address the ranking bias that I observe in my dataset. In other words, the unbiased ranker produces search results where the statistical distributions of ranks for female and male candidates are identical. By construction, this approach eliminates ranking bias, but it may not be fair, because it does not consider candidates' features other than gender.

To implement an unbiased ranker, I split the candidates in the original search results into female and male groups, while preserving the original order. Then I interleave the two groups to reconstruct a new, unbiased rank ordering of the candidates. For example, consider a list of search results \mathbf{R} from a hiring website

$$\mathbf{R} = [(1, m), (2, m), (3, m), (4, f), (5, m), (6, f)],$$

where each tuple in the list is a candidate, and the format of each tuple is $(rank, gender)$. I separate the candidates into female and male lists

$$\mathbf{R}_{\text{female}} = [(4, f), (6, f)]$$

$$\mathbf{R}_{\text{male}} = [(1, m), (2, m), (3, m), (5, m)].$$

Next, I interleave $\mathbf{R}_{\text{female}}$ and \mathbf{R}_{male} back into a single, unbiased list. For each candidate c_i from the smaller list \mathbf{R}_i , I place them in the middle of k candidates $[c_{j1}, c_{j2}, \dots, c_{jk}]$ from the larger list \mathbf{R}_j , where $k = \lceil |\mathbf{R}_i| / |\mathbf{R}_j| \rceil$ and $|\mathbf{R}_i| \leq |\mathbf{R}_j|$. This produces clusters of $k + 1$ candidates of the form $[c_{j1}, c_{j2}, \dots, c_{j(\frac{k}{2})}, c_i, c_{j(\frac{k}{2}+1)}, \dots, c_{jk}]$.⁷ Returning to my example, the new unbiased search results are

$$\mathbf{R}_{\text{merged}} = [(1, m), (4, f), (2, m), (3, m), (6, f), (5, m)].$$

The final step is to reassign the rank values to the candidates according to the new order

$$\hat{\mathbf{R}}_{\mathbf{u}} = [(1, m), (2, f), (3, m), (4, m), (5, f), (6, m)].$$

⁷Another possible approach to producing unbiased rankings would be selecting candidates in-order from R_i and R_j based on weighted random sampling, with weights proportional to $|\mathbf{R}_i|$ and $|\mathbf{R}_j|$. However this approach only produces unbiased rankings in expectation, and may fail in practice if $|\mathbf{R}_i| \ll |\mathbf{R}_j|$. In contrast, my approach is deterministic, and produces unbiased rankings by construction.

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

Note that my unbiased ranker retains the original order of male and female candidates, respectively, as they appeared in the original search results. Thus, within each gender group, candidates are still ordered according to whatever features are used by a given resume search engine. All that the unbiased ranker changes are the positions of female and male candidates relative to the other group.⁸

Neutral Ranker. The goal of my neutral ranker is to address the fairness issues that I observe in my dataset. To achieve gender-agnostic search results, the neutral ranker removes the gender effect from search results that is captured by the *Probability of Being Male* coefficient in my MLMs. Specifically, I can calculate an adjusted rank for each candidate $\hat{\mathbf{R}}_n$ (corresponding the adjusted log rank $\hat{\mathbf{y}}_n$) by subtracting the gender effect from the dependent variable \mathbf{y} :

$$\hat{\mathbf{y}}_n = \mathbf{y} - \mathbf{X}_{\text{gender}}\beta_{\text{gender}},$$

where $\mathbf{X}_{\text{gender}}$ is the column corresponding to gender in \mathbf{X} , and β_{gender} is the gender effect. By re-ranking the candidates according to $\hat{\mathbf{y}}_n$, I am able to produce a gender-agnostic rank for candidates.

In essence, my neutral ranker simply adjusts the original rank of each candidate by removing the “bonus” for being male that I observe in Table 5.5. Thus, the results are fair by construction. Since I only manipulate β_{gender} , the relative ordering of candidates with respect to other features (e.g., experience and education) will not substantially change.

5.4.2 Evaluation

In this section, I evaluate my ranking algorithms. I consider their impact on the rank of female and male candidates, as well as whether their outputs are unbiased and fair.

Ranking Similarity. I begin by evaluating how close the rankings produced by my algorithms are to the original rankings produced by the hiring websites. Figure 5.6 shows the nDCG distributions across job titles and cities for my unbiased and neutral rankers. I plot separate lines for each hiring website, and separate the figures for the top 1000 and top 100 candidates (since these candidates are, arguably, most important to recruiters).

There are three takeaways from Figure 5.6. *First*, I see that the unbiased ranker causes more changes to the search results than the neutral ranker, relative to the original search results. However, *second*, in absolute terms, both algorithms produce results that are similar to the original data: 84-93% of the nDCG scores for the unbiased ranker are ≥ 0.9 across all three websites, while

⁸For simplicity, I do not show candidates with ambiguous gender in this example. In my implementation, I put ambiguous candidates in a separate list $\mathbf{R}_{\text{ambiguous}}$, and merge it with $\mathbf{R}_{\text{female}}$ and \mathbf{R}_{male} during the final steps.

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

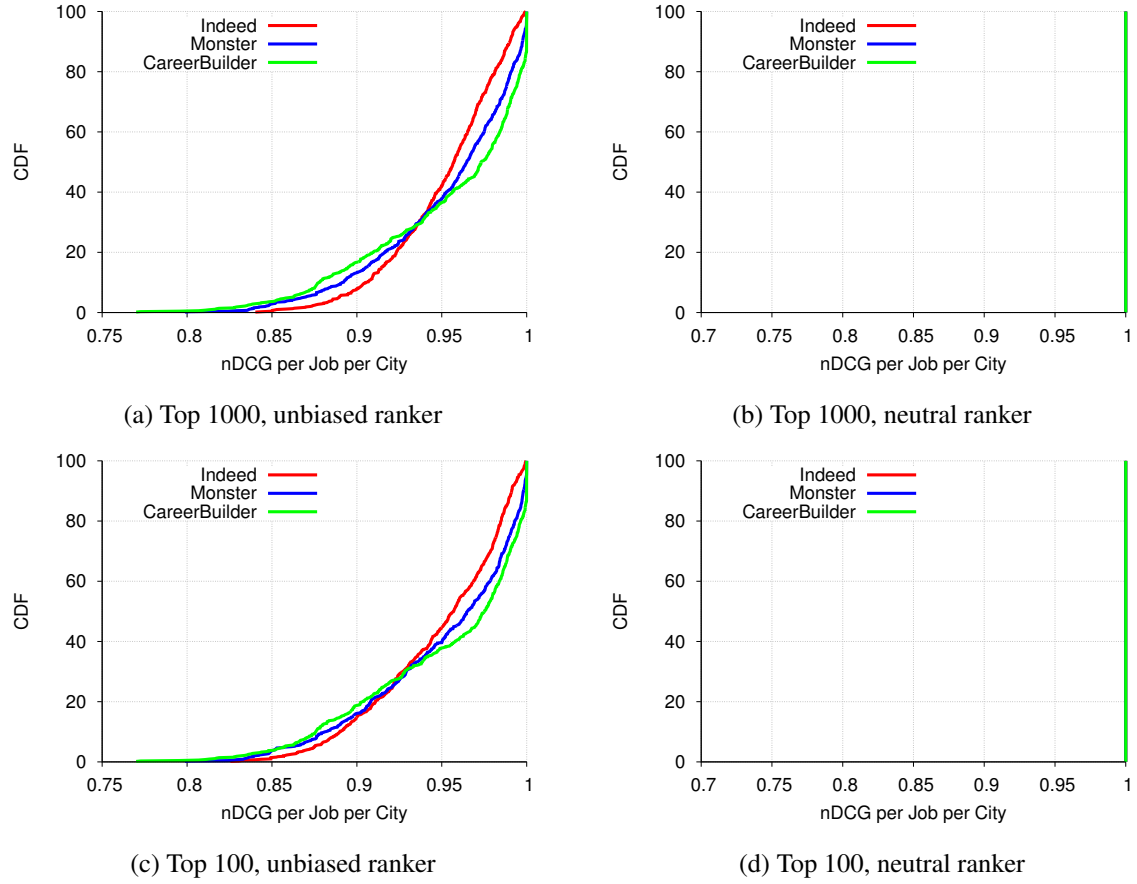


Figure 5.6: nDCG comparison of the unbiased and neutral rankers when compared to the original search results. Figures 5.6a and 5.6b focus on the top 1000 candidates, while Figures 5.6c and 5.6d focus on the top 100.

all of the nDCG scores for the neutral ranker are close to 1. The nDCG scores produced by the neutral ranker make intuitive sense, given that the *Probability of Being Male* feature has a relatively small coefficient on all the hiring sites (see Table 5.5) Finally, the nDCG lines have similar trends when comparing the top 1000 to the top 100 candidates. This shows that my algorithms are robust even when subsampling the original data.

Impact on Ranking Bias. Next, I evaluate the impact of my algorithms on ranking bias, i.e., do they eliminate it? To answer this question, I use the M-W U test to examine the distributions of male and female candidates in the search results produced by the unbiased and neutral rankers. I apply the same rules as when I evaluated the original search results, i.e., I only examine samples with ≥ 20 male and female candidates.

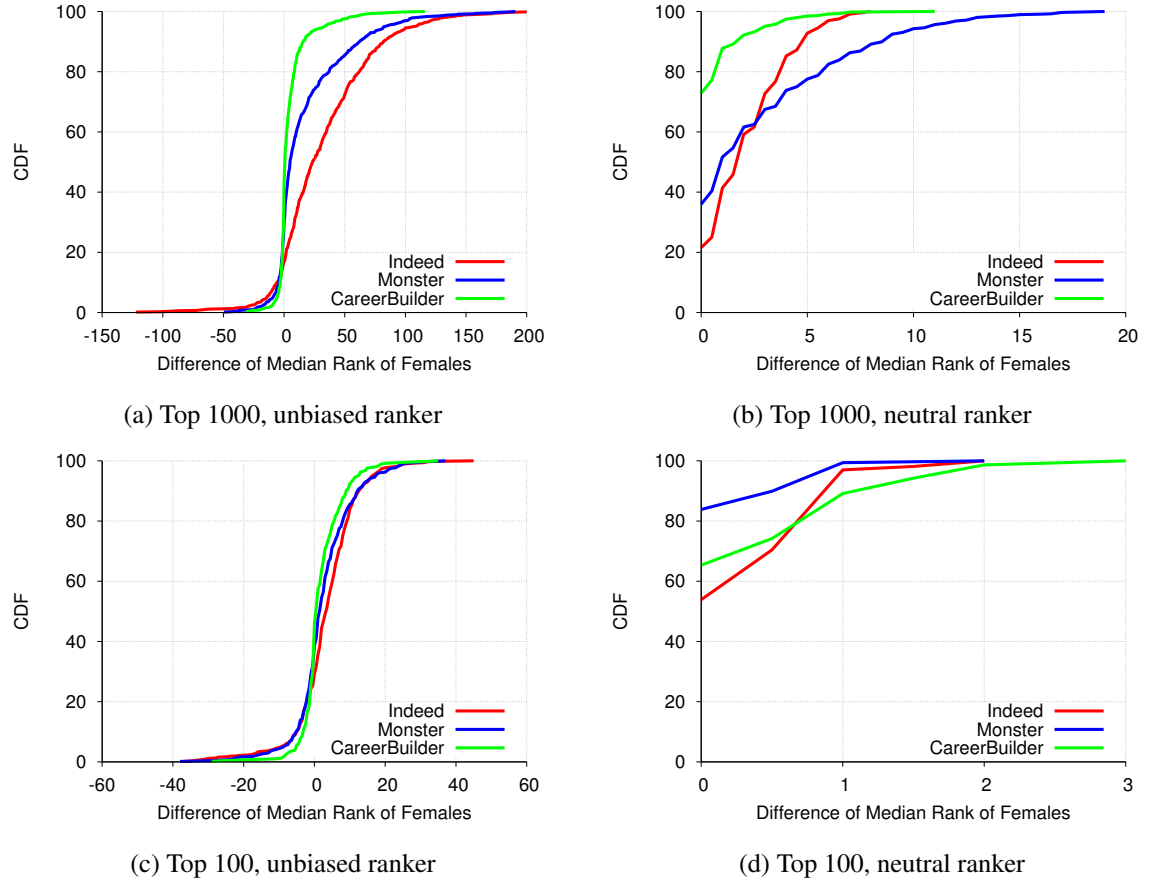


Figure 5.7: Change in the median rank of female candidates after applying my unbiased and neutral rankers. Positive values correspond to increases in the median rank for female candidates after re-ranking. Figures 5.7a and 5.7b focus on the top 1000 candidates, while Figures 5.7c and 5.7d focus on the top 100.

Table 5.4 shows the results of the M-W U tests on search results produced by my unbiased and neutral algorithms. As expected, my unbiased ranker completely removes the ranking bias, because I reorder the female and male candidates evenly. However, for the neutral ranker, ranking bias remains as prevalent as in the original data. This is because the neutral ranker is designed to remove the gender *unfairness*, and the gender coefficient is small in my regression models.

The results in Table 5.4 reveal the first tradeoff between my two ranking algorithms: guaranteeing fair search results does necessarily guarantee unbiased search results.

Impact on Female Candidates and the Gender Gap. The nDCG and M-W U tests show the impact of reordering candidates at the level of whole lists of search results. However, these

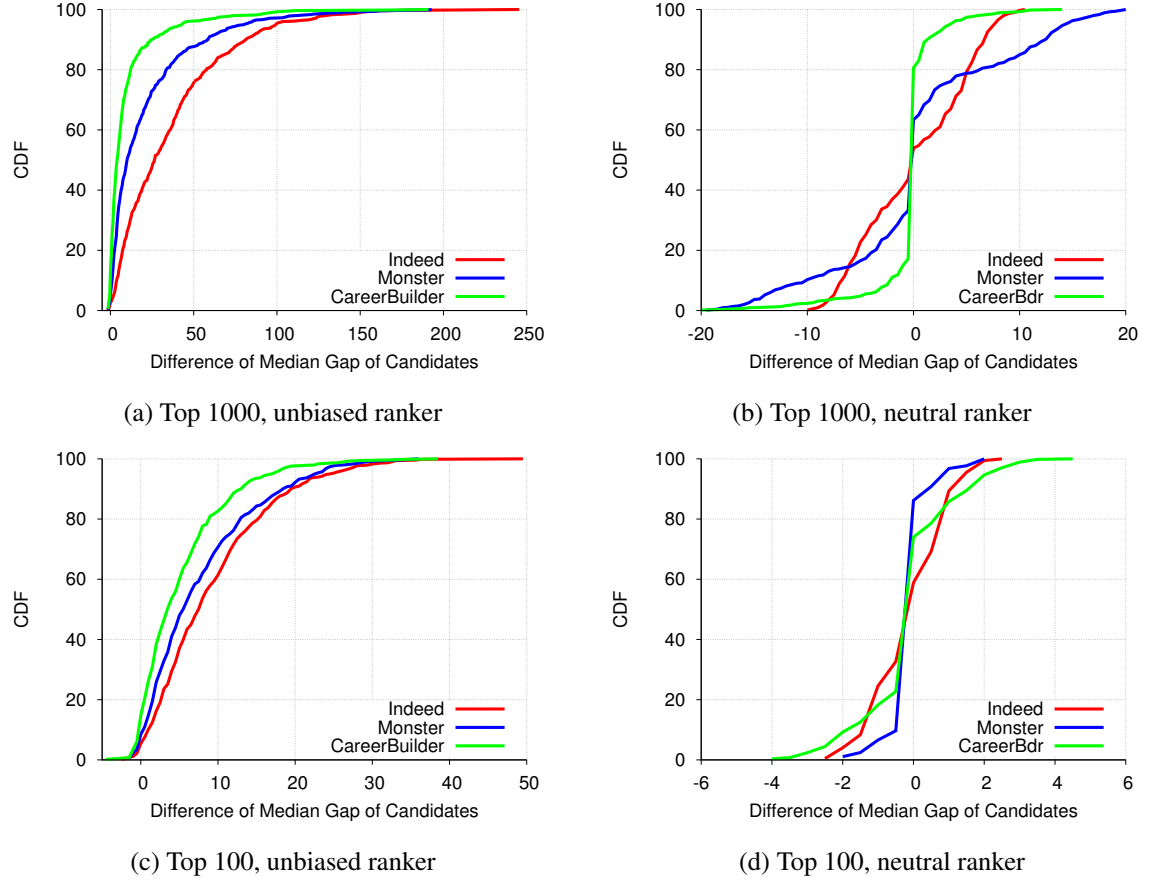


Figure 5.8: Change in the gender gap after applying my unbiased and neutral rankers. Positive values correspond to shrinking gender gaps, i.e., the median rank between male and female candidates is getting smaller after re-ranking. Figures 5.8a and 5.8b focus on the top 1000 candidates, while Figures 5.8c and 5.8d focus on the top 100.

metrics do not tell me about the actual rank changes experienced by candidates, i.e., are female candidates moving up or down, and are they getting closer or farther from male candidates? To answer these questions, I evaluate the following two metrics for each re-ranked list of search results.

First, I examine the change in median rank for female candidates due to my ranking algorithms. I calculate

$$d_f(\mathbf{R}, \hat{\mathbf{R}}) = \tilde{m}_f(\mathbf{R}) - \tilde{m}_f(\hat{\mathbf{R}})$$

where $\tilde{m}_f(\mathbf{R})$ is the median rank of female candidates in the original search results \mathbf{R} , and $\hat{\mathbf{R}}$ are the search results after applying one of my ranking algorithms. Intuitively, if $d_f(\mathbf{R}, \hat{\mathbf{R}})$ is positive,

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

female candidates are moving up in the ranking after applying my ranking algorithm.

Figure 5.7 shows $d_f()$ for each search result sample in my dataset. As before, I have separate lines for each hiring website, and separate plots for the unbiased and neutral rankers, broken down over the top 1000 and top 100 candidates.

There are three takeaways from Figure 5.7. *First*, the unbiased ranker can cause the median rank of female candidates to increase or decrease. This makes sense because there are some job titles (e.g., Registered Nurse) where female candidates have stronger attributes than male candidates, so redistributing the candidates actually lowers the median rank of females in these cases. However, 84%, 70%, and 57% of the top 1000 search result samples experience a positive $d_f()$ on Indeed, Monster, and CareerBuilder, respectively, after applying the unbiased ranker. This shows once again that the majority of search results in my dataset contain ranking bias against female candidates. *Second*, the neutral ranker always increases the median rank of female candidates. This is expected since the neutral ranker removes the negative impact of the *Probability of Being Male* feature, i.e., it lowers the rank of male candidates, effectively pushing up the rank of female candidates. However, the absolute amount of rank improvement for female candidates is marginal due to the small value of the *Probability of Being Male* coefficient. *Finally*, I observe similar trends for the top 1000 and top 100 candidates on all three sites.

Next, I examine the “gap” between male and female candidates in the search results, expressed as the difference in their medians before and after re-ranking the search results using my algorithms. I calculate

$$d_{\text{gap}}(\mathbf{R}, \hat{\mathbf{R}}) = |\tilde{m}_m(\mathbf{R}) - \tilde{m}_f(\mathbf{R})| - |\tilde{m}_m(\hat{\mathbf{R}}) - \tilde{m}_f(\hat{\mathbf{R}})|,$$

where $\tilde{m}_m(\mathbf{R})$ and $\tilde{m}_f(\mathbf{R})$ are the median ranks of male and female candidates, respectively, in the original search results \mathbf{R} . If $d_{\text{gap}}(\mathbf{R}, \hat{\mathbf{R}}) > 0$, then my ranking algorithm is shrinking the rank gap between female and male candidates when comparing the original search results \mathbf{R} to the re-ranked results $\hat{\mathbf{R}}$. Alternatively, if $d_{\text{gap}}(\mathbf{R}, \hat{\mathbf{R}}) = 0$ then the gap is unchanged, and if $d_{\text{gap}}(\mathbf{R}, \hat{\mathbf{R}}) < 0$ then the gap is widening.

Figure 5.8 shows $d_{\text{gap}}()$ for each search result sample in my dataset. Comparing Figures 5.7 and 5.8, I see that the impact of my two ranking algorithms has essentially flipped. The unbiased ranker closes the rank gap in almost all cases on the three platforms (recall that positive values indicate that the gender gap is shrinking). In 84%, 50%, and 26% of search results the gender gap shrinks more than 10 on Indeed, Monster, and CareerBuilder. In contrast, the neutral ranker produces

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

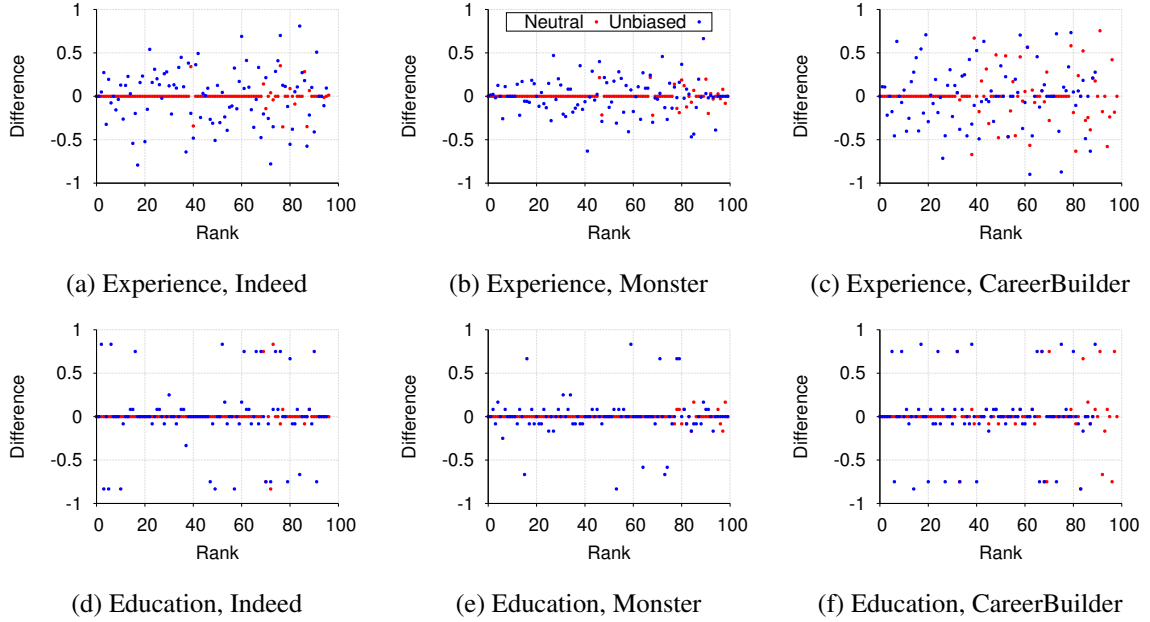


Figure 5.9: Change in the value of the *Experience* and *Education* feature after applying my unbiased and neutral rankers on the top 100 Software Engineer candidates in New York City. Red dots are for the unbiased ranker, while blue dots are for the neutral ranker, as shown in Figure 5.9b.

roughly equal cases of increasing and decreasing gender gaps. I observe similar trends for both the top 1000 and top 100 candidates across the three hiring websites.

Once again, I find that there are tradeoffs when using my ranking algorithms. The unbiased ranker decreases the median rank of female candidates in some job titles and cities, which seems like an undesirable outcome in a society where women are known to face structural barriers and discrimination in employment. However, the unbiased ranker also reduces the gap between the ranks of male and female candidates, which sounds like a desirable outcome from the perspective of promoting gender equality. Conversely, the neutral ranker exhibits the exact opposite properties: it uniformly increases the ranks of female candidates (as it is designed to), but this has the side-effect of sometimes increasing the gender gap in professions where women are historically strong (e.g., Registered Nurses and Cashiers).

Impact on Feature Values. Finally, I examine the impact of my unbiased and neutral rankers on the value of candidate features. The question is: do the candidates at a given rank i have similar features before and after re-ranking the search results using my algorithms? Ideally, I would like the change in value of candidate features before and after re-ranking to be small, since I would

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

like to preserve the ordering function used by the hiring website’s original ranking function. For example, Monster primarily ranks candidates by the relevancy of their job title and their experience (See Table 5.5); this creates reasonably well-organized search results, where candidates with similar features appear close together. However, it might confuse a recruiter and degrade the usability of the search results if candidates with very different features were mixed together, even if this was done in the interest of removing bias or increasing fairness.

To examine the change in feature values, I calculate the difference in value of a given feature x for the candidate at each rank i in search results \mathbf{R} and $\hat{\mathbf{R}}$ (i.e., before and after re-ranking using one of my algorithms). For example, if the candidate at rank 1 originally had 10 years of experience, and the new candidate at rank 1 after re-ranking has 5 years of experience, I would calculate a change of 5 for rank 1.

As a representative example, Figure 5.9 shows the changes in feature values for *Experience* and *Education* of top 100 Software Engineer candidates in New York City on the three hiring websites. Note that I normalize all features in my dataset to be in the range $[0, 1]$, so all differences will be in the range $[-1, 1]$. The red dots correspond to the changes induced by the unbiased ranker, while blue dots are the neutral ranker. I see that the unbiased ranker introduces many changes in the feature values. This is due to the goal of the unbiased ranker, i.e., fixing ranking bias, without controlling for user features other than gender. On the other hand, the neutral ranker does not change the feature values of candidates most of the time, because 1) it makes fewer changes to the ranking overall (compared to the unbiased ranker), and 2) when it does make changes, it tends to move candidates relatively short distances (see Figure 5.7).

The takeaway from Figure 5.9 is that the pattern of tradeoffs between my ranking algorithms continues. If the designer of a resume search engine desires to create search results that follow some relatively deterministic ordering (e.g., by experience or resume update time), the neutral ranker is superior: it ensures gender fairness, but also allows the hiring website to control for other candidate features. This may be a desirable property from a usability perspective. Conversely, to remove ranking bias, the unbiased ranker needs the flexibility to dramatically reorder candidates, which makes it difficult to create search results that are organized around a logical ordering.

5.5 Summary

In this study, I examine gender inequality on the resume search engines provided by Indeed, Monster, and CareerBuilder. I crawled search results for 35 job titles across 20 U. S. cities; these

CHAPTER 5. IMPACT OF GENDER ON HIRING SITES

contain data on 855K candidates. Using statistical tests, I find two significant issues with the search results produced by these search engines:

- **Ranking Bias:** 18.2%, 13.5%, and 18.2% of job title/city pairs on Indeed, Monster, and CareerBuilder (respectively) show statistically significant ranking bias. This corresponds to between 6 and 16 distinct job titles (depending on the hiring website) where male candidates appear at systematically higher ranks in search results than female candidates.
- **Unfairness:** I find statistically significant, negative correlations between rank and gender in my dataset. This means that even when controlling for all other visible candidate features, there is a slight penalty against female candidates.

My two metrics for measuring gender inequality (ranking bias and fairness) are complementary. Search results that exhibit ranking bias are not necessarily unfair, since the former metric does not control for candidate features other than gender. Only by examining both metrics can I obtain a thorough understanding of gender inequality in search results.

Using controlled experiments, I probe the hidden features that influence candidate rank on these resume search engines. Importantly, **I find that none of the three websites directly use gender as a feature**, i.e., their algorithms are not explicitly discriminatory. This concurs with the design of the hiring websites, which do not ask for candidates to input their gender. However, I do see that other hidden features (unemployment and alma mater) are taken into account. I hypothesize that these hidden features may be the underlying cause of the unfairness I observe in search results, since one or more of these features may correlate with candidate's gender.

Finally, analysis of my proposed ranking algorithms highlights the tradeoffs that are inherent when enforcing different definitions of fairness. My unbiased algorithm is effective at reducing the gap between the median rank of male and female candidates, which can be viewed as a positive step towards gender equality. On the other hand, it reduces the median rank of female candidates in some job titles (e.g., Registered Nurse); it also produces search results that intermingle candidates with very different qualifications at similar ranks, which may confuse recruiters and impede usability. My neutral ranker offers *the exact opposite* outcomes: it uniformly improves the rank of female candidates, and produces nicely sorted search results, but it can increase the gender gap in certain circumstances.

Chapter 6

Conclusion

In §3, §4, and §5 I described my previous work in detail. In this section, I summarize the key findings and impact of my work. I also discuss my procedure for conducting ethical algorithmic auditing studies. Finally, I provide general guidelines to future studies focusing on algorithm audits.

6.1 Summary and Impact

My thesis examined three opaque systems: the ride-sharing system Uber, the e-commerce marketplace Amazon, and the online hiring websites Indeed, Monster, and CareerBuilder. The findings of my work reveal fairness and effectiveness issues on these systems.

Uber Study. In the Uber study, I present the first in-depth analysis of Uber. I leverage four weeks of data collected from Uber’s smartphone app and official API that covers midtown Manhattan and downtown San Francisco. Using the measured data on supply, demand, surge multipliers, etc., I am able to uncover some implementation details of Uber’s dynamic pricing algorithm. In particular, Uber divides cities into separate surge regions, where each region independently updates its surge multiplier on a 5-minute clock, based on the supply and demand in the immediately previous 5-minute window. Leveraging the surge regions, I propose a method to potentially reduce the price for Uber riders: walk to a neighboring surge region to receive a lower surge multiplier. My experiment shows that this method can save customers 50% on fares 20% of time when the rider is around Times Square.

In addition, my findings reveal that the surge pricing mechanism may not be as effective as Uber claims. Uber has stated that the goals of surge pricing are to increase supply and intentionally reduce demand, and they claim that the system increased the number of drivers by 70-80% after

CHAPTER 6. CONCLUSION

it was introduced [84]. However, as I show in §3, while surges do seem to have a small effect on attracting new cars, it also causes drivers to either become idle or even drive away from the surge area. This demonstrates that the surge mechanism may not be as effective at attracting driver supply as it is at depressing passenger demand.

More importantly, my investigation discovered a consistency bug that was serving random surge multipliers to customers. I notified Uber of this bug and they fixed it. However, the bug had existed for 6 months without being noticed by Uber, causing potential harm to customers.

Amazon Study. In this study, I took the first steps towards detecting and quantifying sellers using algorithmic pricing on Amazon Marketplace. I collected four months of data on over 1600 products and 22000 sellers on Amazon marketplace, and detected over 500 algorithmic sellers using my detection methodology.

My findings uncover the power of algorithmic sellers in Amazon marketplace. Sellers I identified as using algorithmic pricing receive more feedback and win the Buy Box more frequently. My data-driven simulations suggest that algorithmic sellers have higher sales volumes and thus more revenue than non-algorithmic sellers, under a variety of different customer purchasing models. Receiving more feedback may create a feedback loop for algorithmic sellers to generate higher sales volumes, since feedback is a key feature used by the Buy Box algorithm. This makes it difficult for non-algorithmic sellers to compete in the market.

Hiring Study. In this study, I examine gender inequality on the resume search engines provided by Indeed, Monster, and CareerBuilder. I crawled search results for 35 job titles across 20 U.S. cities; these contain data on 855K candidates. Using statistical tests, I find two significant issues with the search results produced by these search engines: female candidates are unfairly ranked lower in general according to my two equality definitions, *Ranking Bias* and *Unfairness*, as described in §5.

To combat the inequality, I propose two alternative ranking methods that separately address these two scenarios. The evaluation of the two new rankers highlights the tradeoffs that are inherent in operationalizing different definitions of fairness: 1) guaranteeing fair search results does not necessarily guarantee unbiased search results; 2) my unbiased ranker reduces the gender gap but may lower female rank in some cases, whereas my neutral ranker always increases female rank but may enlarge the gender gap; 3) my unbiased ranker reorders the candidates dramatically, which may produce search results that are not intuitively ordered for recruiters.

Impact of my Work. My work has practical ramifications for consumers, and consequently it has received significant media attention in venues such as NBC News, ABC

CHAPTER 6. CONCLUSION

News, the Boston Globe, NPR, USA Today, Fortune, Money, Consumer Affairs, and Motherboard [40, 66, 87, 99, 108, 115, 117, 125, 143, 146, 165, 185]. This has helped raise awareness of algorithmic issues with the public, and pressure companies to become more transparent. For example, Uber started a new transparency blog in response to my Uber study [127].

I am one of the few scholars that has empirically studied platform marketplaces using an observational approach. This has attracted the attention of scholars from many research disciplines, including: economics, statistics, civil engineering, and computer science. By sharing my methodology, code, and data with these research groups, we are working together toward building more transparent, efficient, and fair marketplaces for people. I am honored that I was invited to publish a letter in ACM SIGEcom Exchanges highlighting my work on Uber and Amazon [53]. My empirical work is also heavily cited in ongoing litigation against Uber [1].

I also collaborate with governmental regulators such as the European Commission and a transportation commission in a major U.S. city. By adopting my measurement methodology, the European Commission was able to identify and formally challenge price discrimination by the Disneyland theme park in Europe. This led to Disneyland changing its business practices.

Finally, as my contribution to researchers who will be working on these topics in the future, I share my code and data at

<http://personalization.ccs.neu.edu>

6.2 Ethics for Algorithm Audits

Conducting an ethical algorithmic audit of an opaque system requires extra care to minimize the impact and harm on all involved parties: the users of that system, and the company that runs the system. In this section, I describe my procedure for conducting ethical algorithmic audits that I follow in my studies, which I hope will serve as a reference for future investigators.

1. To initiate an algorithmic auditing study on a service, the first step is to read the service's Terms of Service (ToS), and carefully weigh the risks of gathering data from the service. Consider that gathering data may require crawling, creating fake accounts, or borrowing the accounts of real users (e.g., through an in-person lab experiment, or via crowdsourcing). All of these methods may violate the company's ToS, which some courts have construed as violations of the Computer Fraud and Abuse Act (CFAA). If the legal risks from potentially breaking the CFAA are beyond what the investigator can comfortably manage, the study has to be abandoned, or

CHAPTER 6. CONCLUSION

alternative strategies for data collection must be devised. Fortunately, researchers are fighting to update this outdated and vague law [16], and their effort has already shown some positive changes [119].

2. Next, the investigator needs to consult an Institutional Review Board (IRB) to get an approval or a waiver to start the study, especially when the service under investigation involves human users. Audits are a form of deception study, where the company or service is the subject, and they are unaware that an experiment is being conducted. As with any deception study, the investigator should make plans to debrief the subject at the conclusion of the experiment. Furthermore, the investigator should minimize their impact on human users of the target service; ideally, the experimental methodology should not directly interact with users in any way (i.e., during data collection). Finally, the investigator should carefully consider what data is necessary for their analysis, and minimize collection of personal information, or obfuscate this data after collection.
3. Once the study is approved by the IRB, the investigator needs to decide the rate and scale of the data crawler. The investigator should minimize the impact of the crawler on the service: set the crawling rate just enough to get sufficient data for the study. When the service requires creating accounts to scale the data collection, the investigator should create a minimal amount of fake accounts to conduct the study, and (if possible) delete them after collection is complete.
4. Before the research work is publicly released (e.g., when it gets accepted to an academic venue), the investigator should contact the audited companies to responsibly disclose the findings. This gives the company an opportunity to correct any factual errors in the study, and it is appropriate to allow the company to add a statement to the publication if they so desire.
5. Finally, the investigator may share the data with other research groups, subject to IRB restrictions and privacy norms. Researchers should take care to ensure that users' contextual privacy is preserved when deciding whether to release data [140]. The release of data represents a fundamental tradeoff between facilitating reproducibility, versus protecting user's privacy, and the researcher should carefully consider how to manage this issue. Nonetheless, the investigator can always share their code and methodologies to peers to help reproduce the work and facilitate future studies.

6.3 A Guide for Future Audits

In this section, I share the lessons I learned about how to choose systems to audit. Although there are an increasing number of deployed systems that utilize opaque algorithms, not all of them are equally interesting and worthy of researchers time. In particular, I discuss two issues that may make a system biased and harmful to users, and thus worth investigating using an algorithm audit: *Incentive Misalignment* and *Biases in Human Data*.

6.3.1 Incentive Misalignment

Misaligned incentives occur when the participants on a platform have different and conflicting goals [36, 138]. For example, in a marketplace, sellers are incentivized to increase the product price for more profit, but buyers want to decrease the price for more savings. This is not problematic for open marketplaces as they tend to reach equilibrium naturally. However, for marketplaces that use opaque algorithm to control the price, the misaligned incentives may result in *unfairness* and *manipulation*. In general, any system that has the potential for misaligned incentives is a potential target for an audit.

Unfairness. The incentive for almost all businesses is to profit. However, sometimes the methods used to obtain profit may introduce unfairness to users. This is particularly problematic in opaque systems, such as Uber and Amazon, where there is information asymmetry (i.e., the platform provider has information that is unavailable to customers and sellers) or structural barriers (i.e., the platform provider has control over how information is presented to users).

On Uber, information about the supply of vehicles and demand for rides are not available to customers, and only indirectly available to drivers (presented as a heatmap of surges throughout an area). Uber has been accused of using multiple “tricks” to increase profits that rely on this information asymmetry. These instances include: upfront pricing showing different routes and fares for drivers and riders [183]; a price fixing scheme is currently being litigated [43]; and possible exploitation of Uber users being more likely to pay surge prices when their phone battery is low [56]. Although the pricing bug I discovered in §3 is not discriminatory pricing, it easily could have been, and nobody would have known if I had not audited Uber’s systems.

Amazon has enormous power to steer the buying habits of its customers by controlling the order of products in search results, as well as choosing which seller occupies the Buy Box. As I show in §4, the “is Amazon?” feature has some weight in deciding the winner of the Buy Box, which may

CHAPTER 6. CONCLUSION

indicate that Amazon favors themselves to win the Buy Box (thus increasing their own profit at the expense of third-party sellers). I also show that the Buy Box tends to highlight sellers that do not offer the lowest prices on goods; although customers can still access the lowest priced offers, this requires additional clicks and searching, which creates a powerful disincentive against this behavior.

Manipulation. Opaque systems are vulnerable to manipulation: once the participants learn the systems, they may be able to “game” it. For example, drivers on Uber exploited the surge pricing algorithm by intentionally colluding to reduce supply and this induce surge pricing [19]. Similarly, sellers on e-commerce marketplaces can exploit the feedback feature to boost their reputation among competitors [190].

Manipulation is also possible on the customer side. As I present in §3, Uber riders are able to avoid surge pricing by walking to a neighboring surge region. This benefits savvy riders, but reduces profits for Uber and drivers. In another infamous example, customers were able to exploit Walmart’s automatic price matching system by listing Playstation 4 consoles on Amazon for below-market prices, then purchasing the consoles from Walmart after they dropped their price [79]. In both of these cases, knowledge of the system’s mechanisms allows people to manipulate them in ways the designers had not considered or accounted for.

6.3.2 Biases in Human Data

Human beings have cognitive biases that influence the data they create. In essence, human biases become embedded in data, making it non-neutral with respect to sensitive issues like gender, race, etc. These biases make it challenging for system designers to leverage data collected from human beings: unless the algorithms are very carefully designed, the biases in the data may train the algorithms to exhibit unfairness. Any system that relies on data from human beings to power a sensitive procedure is a valid potential target for an algorithm audit.

For example, on online hiring websites, if the click-through rate is used as a feature for ranking candidates, and the recruiters exhibit (unconscious) bias towards people with certain demographic traits (e.g., women and/or minorities), then the impacted sub-populations will be unfairly ranked lower, harming their opportunities for being seen, interviewed, and hired. Similarly, prior work has shown that in online gig-economy markets, customers leave biased feedback and reviews for workers, which then led to search results that were also biased against these marginalized workers [90]. Finally, researchers have shown that racial discrimination exists in other marketplaces such as eBay [20] and AirBnb [69].

CHAPTER 6. CONCLUSION

Even worse, algorithms that are biased by human data may create feedback loops that continuously harm the marginalized sub-population. For example, on the hiring websites, if candidates with certain demographic traits are less likely to be hired, they will have less working experience than other competing peers. As I show in §5, *experience* is a significant feature affecting the candidate's ranking, so less *experience* translates to lower rank, making the candidate even less likely to be hired in the future. Likewise, in the online gig-economy, biased feedback from customers may hurt workers' ratings, making the workers less likely to get hired again in the future. This pernicious cycle can be thought of as a new form of algorithmically-driven structural bias.

To address these issues, one option is to develop fair Machine Learning algorithms that control for human bias within datasets [76, 85, 92]. This method is similar to my **Neutral Ranker** in §5. However, as I showed, this method creates tradeoffs that may not satisfy all notions of fairness. Further, if some highly predictive feature is not adopted due to potential biases against certain demographics, the predictive power of the algorithms will be weakened. This raises fundamental questions about the applicability of predictive algorithms, i.e., there may be scenarios where accurate prediction is entirely reliant on stereotypes and structural biases that exist in reality, but are undesirable from a social perspective. Finally, although new tools or techniques may address bias issues in algorithms, we must still rely on algorithmic auditing to verify that deployed systems are using these new tools and obeying best practices. Meanwhile, system designers need to be vigilant about how their platforms, content, and the behavior of users change over time, as people are inherently unpredictable, and bias can re-emerge in unexpected ways.

Bibliography

- [1] Meyer v. Kalanick, no. 1:2015cv09796 - document 37 (S.D.N.Y. 2016).
- [2] Prime time for drivers. Lyft. <https://help.lyft.com/hc/en-us/articles/214586017-Prime-Time-for-Drivers>.
- [3] What is surge pricing? Uber. <https://help.uber.com/h/6c8065cf-5535-4a8b-9940-d292ffdce119>.
- [4] Discrimination of consumers in the digital single market. European Parliament, 2013.
- [5] Amazon marketplace web service (Amazon MWS) documentation. Amazon.com, 2015. http://docs.developer.amazonservices.com/en_US/dev_guide/index.html.
- [6] Amazon marketplace web service (Amazon MWS) documentation – submitfeed. Amazon.com, 2015. http://docs.developer.amazonservices.com/en_US/feeds/Feeds_SubmitFeed.html.
- [7] Amazon sellers sold record-setting more than 2 billion items worldwide in 2014. Business Wire, January 2015. <http://www.businesswire.com/news/home/20150105005186/en/Amazon-Sellers-Sold-Record-Setting-2-Billion-Items>.
- [8] Fees and pricing. Amazon.com, 2015. <https://www.amazon.com/gp/help/customer/display.html?nodeId=1161240>.
- [9] Former e-commerce executive charged with price fixing in the antitrust division's first online marketplace prosecution. U.S. Department of Justice, April 2015. <http://bit.ly/2oBjSvn>.

BIBLIOGRAPHY

- [10] Fulfillment fees for orders on amazon.com. Amazon.com, 2015. <http://www.amazon.com/gp/help/customer/display.html/?nodeId=201119410>.
- [11] Increase your chances of winning the buy box. Amazon.com, 2015. https://www.amazon.com/gp/help/customer/display.html/ref=hp_rel_topic?ie=UTF8&nodeId=201687830.
- [12] Keeping an eye on google - eye tracking serps through the years. 2015.
- [13] Match low price. Amazon.com, 2015. http://www.amazon.com/gp/help/customer/display.html/ref=hp_left_cn?ie=UTF8&nodeId=200832850.
- [14] The programmatic-advertising report: Mobile, video, and real-time bidding drive growth in programmatic. Business Insider, 2015. <http://read.bi/1Eb6OBd>.
- [15] Big data: A tool for inclusion or exclusion? Federal Trade Commission, Jan. 2016.
- [16] Challenge to cfaa prohibition on uncovering racial discrimination online. ACLU, 2016. <https://www.aclu.org/legal-document/sandvig-v-lynch-complaint>.
- [17] Ranking of emplyment sites. Alexa, 2016. <http://www.alexa.com/topsites/category/Business/Employment>.
- [18] J. Angwin and D. Mattioli. Coming soon: Toilet paper priced like airline tickets. The Wall Street Journal, 2012. <http://www.wsj.com/articles/SB10000872396390444914904577617333130724846>.
- [19] Anonymous. Empty promises. Confessions of an Uber Driver Blog, Jan. 2015. <http://bit.ly/1c4JZmZ>.
- [20] I. Ayres, M. Banaji, and C. Jolls. Race effects on eBay. *The RAND Journal of Economics*, 46(4):891–917, 2015.
- [21] M. Babaioff, B. Lucier, and N. Nisan. Bertrand networks. In *Proceedings of the Fourteenth ACM Conference on Electronic Commerce*, EC ’13, 2013.
- [22] M. Babaioff, N. Nisan, and R. Paes Leme. Price competition in online combinatorial markets. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW ’14, 2014.

BIBLIOGRAPHY

- [23] M. Babaioff, R. Paes Leme, and B. Sivan. Price competition, fluctuations and welfare guarantees. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, EC '15, 2015.
- [24] L. Backstrom. News feed fyi: A window into news feed. facebook business, August 2013. <https://www.facebook.com/business/news/News-Feed-FYI-A-Window-Into-News-Feed>.
- [25] M. Backus, T. Blake, D. V. Masterov, and S. Tadelis. Is sniping a problem for online auction markets? In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, 2015.
- [26] N. Baird and P. Rosenblum. Tough love: An in-depth look at retail pricing practices. *RSR Pricing Benchmark*, 2013.
- [27] E. Bakshy, S. Messing, and L. Adamic. Exposure to ideologically diverse news and opinion on facebook. *Science*, 2015.
- [28] S. Barocas and A. D. Selbst. Big data's disparate impact. *104 California Law Review*, 671, 2016.
- [29] M. A. Bashir, S. Arshad, W. Robertson, and C. Wilson. Tracing information flows between ad exchanges using retargeted ads. In *25th USENIX Security Symposium (USENIX Security 16)*, Austin, TX, 2016.
- [30] M. Bendick, C. W. Jackson, and V. A. Reinoso. Measuring employment discrimination through controlled experiments. *The Review of Black Political Economy*, 23(1):25–48, 1994.
- [31] T. Benmayer. The buy box cheat sheet. Feedvisor, May 2014. <http://feedvisor.com/the-buy-box-cheat-sheet/>.
- [32] J. Bertrand. Book review of theorie mathematique de la richesse sociale and of recherches sur les principes mathematiques de la theorie des richesses. *J des Savants*, 67(2):499–508, 1883.
- [33] M. Bertrand and S. Mullainathan. Are emily and greg more employable than lakisha and jamal? a field experiment on labor market discrimination. *The American Economic Review*, 94(4):991–1013, 2004.

BIBLIOGRAPHY

- [34] T. Bishop. Amazonfresh delays \$299/year prime fresh rollout, tweaks subscription model in california. Geek Wire, June 2015. <http://bit.ly/2ojqPAZ>.
- [35] M. Boland. Apple Pay's Real Killer App: The Uber-ification Of Local Services. Huffington Post, January 2015. <http://huff.to/1IxEE3M>.
- [36] L. Boni and K. L. Womack. Wall Street's Credibility Problem: Misaligned Incentives and Dubious Fixes? Center for financial institutions working papers, Wharton School Center for Financial Institutions, University of Pennsylvania, 2002.
- [37] E. Bozdag. Bias in algorithmic filtering and personalization. *Ethics and Inf. Technol.*, Sept. 2013.
- [38] M. E. Brashears. Sex, society, and association: A cross-national examination of status construction theory. *Social Psychology Quarterly*, 71(1):7–85, 2008.
- [39] M. Brohan. Amazon builds up its european marketplace. Internet Retailer, April 2015. <https://www.internetretailer.com/2015/04/30/amazon-builds-its-european-marketplace>.
- [40] K. V. Brown. How to not get screwed on amazon. Fusion, 2016. <http://bit.ly/2phXrKu>.
- [41] Q. BUI. The most common jobs for the rich, middle class and poor. NPR, October 2014. <http://n.pr/2iSViU0>.
- [42] Careerbuilder search. CareerBuilder, 2016. <https://hiring-assets.careerbuilder.com/media/attachments/careerbuilder-original-2660.pdf?1474569898>.
- [43] Z. Carter. The legal problem that could crash uber. Huffington Post, 2017. http://www.huffingtonpost.com/entry/legal-problem-could-crash-uber_us_5718d485e4b0479c59d714f6.
- [44] M. Castillo. Amazon hoax coupled with walmart's price matching leads to ridiculously cheap ps4s. Adweek, November 2014. <http://bit.ly/1t3nD6L>.
- [45] Careerbuilder about us. Careerbuilder, 2016. <http://www.careerbuilder.com/share/aboutus/>.

BIBLIOGRAPHY

- [46] Careerbuilder search. Careerbuilder, 2016. <https://hiring.careerbuilder.com/talentstream/sourcing-technology/search>.
- [47] Most common job titles posted online. CEB, August 2011. <http://bit.ly/2iSYb9h>.
- [48] T. M. Chavous, D. Rivas-Drake, C. Smalls, T. Griffin, and C. Cogburn. Gender matters, too: the influences of school racial discrimination and racial identity on academic engagement outcomes among african american adolescents. *Developmental psychology*, 44(3):637, 2008.
- [49] S. Chawla and F. Niu. The price of anarchy in bertrand games. In *Proceedings of the 10th ACM Conference on Electronic Commerce, EC '09*, 2009.
- [50] S. Chawla, F. Niu, and T. Roughgarden. Bertrand competition in networks.
- [51] L. Chen, A. Mislove, and C. Wilson. Peeking Beneath the Hood of Uber. In *Proceedings of the 15th ACM/USENIX Internet Measurement Conference (IMC'15)*, Tokyo, Japan, October 2015.
- [52] L. Chen, A. Mislove, and C. Wilson. An Empirical Analysis of Algorithmic Pricing on Amazon Marketplace. In *Proceedings of International World Wide Web Conference (WWW'16)*, Montreal, Canada, Apr 2016.
- [53] L. Chen and C. Wilson. Observing algorithmic marketplaces in-the-wild. *SIGecom Exch.*
- [54] Y. Chen, P. Berkhin, B. Anderson, and N. R. Devanur. Real-time bidding algorithms for performance-based display ad allocation. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '11*, 2011.
- [55] J. Chevalier and A. Goolsbee. Measuring prices and price competition online: Amazon. com and barnesandnoble. com. *Quantitative marketing and Economics*, 2003.
- [56] A. Chowdhry. Uber: Users are more likely to pay surge pricing if their phone battery is low. *Forbes*, 2016. <https://www.forbes.com/sites/amitchowdhry/2016/05/25/uber-low-battery>.
- [57] L. Clark. Uber denies researchers' 'phantom cars' map claim. *Wired*, July 2015. <http://www.wired.co.uk/news/archive/2015-07/28/uber-cars-always-in-real-time>.

BIBLIOGRAPHY

- [58] E. K. Clemons, I.-H. Hann, and L. M. Hitt. Price dispersion and differentiation in online travel: An empirical investigation. *Management science*, 2002.
- [59] E. G. Cohen. Expectation states and interracial interaction in school settings. *Annual Review of Sociology*, 8(1):209–235, 1982.
- [60] G. W. Corder and D. I. Foreman. *Nonparametric statistics: A step-by-step approach*. John Wiley & Sons, 2014.
- [61] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. 2008.
- [62] M. L. Cummings. Automation bias in intelligent time critical decision support systems. In *AIAA 3rd Intelligent Systems Conference*, 2004.
- [63] A. Datta, M. C. Tschantz, and A. Datta. Automated experiments on ad privacy settings: A tale of opacity, choice, and discrimination. 2015.
- [64] A. Datta, M. Tschantz Carl, and A. Datta. Automated experiments on ad privacy settings a tale of opacity, choice, and discrimination. In *Proceedings on Privacy Enhancing Technologies 2015*, PoPETs ’14, 2015.
- [65] R. N. David L. Word, Charles D. Coleman and R. Kominski. Demographic aspects of surnames from census 2000. census.gov, 2000. <https://www2.census.gov/topics/genealogy/2000surnames/surnames.pdf>.
- [66] L. Deutsch. How to hack uber surge-charge fares by 10-20%. USA Today, 2015. <https://usat.ly/2pTdwrV>.
- [67] N. Diakopoulos. How uber surge pricing really works. The Washington Post, Apr. 2015. <http://wapo.st/1yBf3EO>.
- [68] T. Duryee. Amazon adds 30 million customers in the past year. Geek Wire, May 2014. <http://www.geekwire.com/2014/amazon-adds-30-million-customers-past-year/>.
- [69] B. G. Edelman, M. Luca, and D. Svirsky. Racial discrimination in the sharing economy: Evidence from a field experiment, 2015. <http://ssrn.com/abstract=2701902>.

BIBLIOGRAPHY

- [70] Eeo-1 job classification guide 2010. U.S. Equal Employment Opportunity Commission, 2010.
<https://www.eeoc.gov/employers/eeo1survey/jobclassguide.cfm>.
- [71] M. M. EL-BERMAWY. Your filter bubble is destroying democracy. Wired, November 2016. <https://www.wired.com/2016/11/filter-bubble-destroying-democracy/>.
- [72] M. Eslami, A. Aleyasen, K. Karahalios, K. Hamilton, and C. Sandvig. Feedvis: A path for exploring news feed curation algorithms. In *Proceedings of the 18th ACM Conference Companion on Computer Supported Cooperative Work & Social Computing*, 2015.
- [73] L. M. Finkelstein, M. J. Burke, and M. S. Raju. Age discrimination in simulated employment contexts: An integrative analysis. *Journal of Applied Psychology*, 80(6):652, 1995.
- [74] K. Fiscella and A. M. Fremont. Use of geocoding and surname analysis to estimate race and ethnicity. *Health Serv Res*, 2006.
- [75] Y. Ge, C. R. Knittel, D. MacKenzie, and S. Zoepf. Racial and gender discrimination in transportation network companies. Technical report, National Bureau of Economic Research, 2016.
- [76] S. Goel, M. Perelman, R. Shroff, and D. A. Sklansky. Combatting police discrimination in the age of big data. *New Criminal Law Review: In International and Interdisciplinary Journal*, 2017.
- [77] W. D. Gouvier, S. Sytsma-Jordan, and S. Mayville. Patterns of discrimination in hiring job applicants with disabilities: The role of disability type, job complexity, and public contact. *Rehabilitation Psychology*, 48(3):175, 2003.
- [78] L. A. Granka, T. Joachims, and G. Gay. Eye-tracking analysis of user behavior in WWW search. 2004.
- [79] K. B. Grant and S. Whitten. Wal-mart closes loophole on playstation 4 scam. CNBC, 2014. <http://www.cnbc.com/2014/11/19/wal-mart-scam-extends-beyond-playstation-4.html>.
- [80] A. Griswold. Does uber’s surge pricing take unfair advantage of drunk people? Slate, 2014.
http://www.slate.com/blogs/moneybox/2014/11/04/uber_price_

BIBLIOGRAPHY

surging_on_halloween_does_it_take_unfair_advantage_of_drunk_people.html.

- [81] A. Griswold. Does Uber’s Surge Pricing Take Unfair Advantage Of Drunk People? Slate, November 2014. <http://slate.me/10s0zXH>.
- [82] Z. Guan and E. Cutrell. An eye tracking study of the effect of target rank on web search. 2007.
- [83] S. Guha, B. Cheng, and P. Francis. Challenges in measuring online advertising systems. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, IMC ’10*, 2010.
- [84] B. Gurley. A deeper look at uber’s dynamic pricing model. Above the Crowd, Mar. 2014. <http://bit.ly/1fmNgI1>.
- [85] S. Hajian, J. Domingo-Ferrer, A. Monreale, D. Pedreschi, and F. Giannotti. Discrimination- and privacy-aware patterns. *Data Min. Knowl. Discov.*, 2015.
- [86] J. V. Hall and A. B. Krueger. An analysis of the labor market for uber’s driver-partners in the united states, Jan. 2015.
- [87] S. Halzack. Looking for the lowest prices on amazon? you may have to dig a little. Washington Post, 2016. <http://wapo.st/2qOQTnx>.
- [88] A. Hannak, P. Sapiezzyński, A. M. Kakhki, B. Krishnamurthy, D. Lazer, A. Mislove, and C. Wilson. Measuring Personalization of Web Search. In *Proceedings of the Twenty-Second International World Wide Web Conference (WWW’13)*, Rio de Janeiro, Brazil, May 2013.
- [89] A. Hannak, G. Soeller, D. Lazer, A. Mislove, and C. Wilson. Measuring price discrimination and steering on e-commerce web sites. 2014.
- [90] A. Hannak, C. Wagner, D. Garcia, A. Mislove, M. Strohmaier, and C. Wilson. Bias in Online Freelance Marketplaces: Evidence from TaskRabbit and Fiverr. 2017.
- [91] A. Hannák, C. Wagner, D. Garcia, A. Mislove, M. Strohmaier, and C. Wilson. Bias in online freelance marketplaces: Evidence from taskrabbit and fiverr. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing, CSCW ’17*. ACM, 2017.

BIBLIOGRAPHY

- [92] M. Hardt, E. Price, , and N. Srebro. Equality of opportunity in supervised learning. pages 3315–3323, 2016.
- [93] B. Helm. The amazon marketplace seller that says it has “the colombian cocaine of algorithms”. Slate, 2016. http://www.slate.com/blogs/moneybox/2016/02/26/pharmapacks_makes_over_70_million_selling_miscellaneous_items_on_amazon.html.
- [94] A. Hern. Angry about facebook censorship? wait until you hear about the news feed. The Guardian, May 2016. <https://www.theguardian.com/technology/2016/may/11/facebook-censorship-news-feed-trending-topics>.
- [95] D. E. Ho, K. Imai, G. King, and E. A. Stuart. Matching as nonparametric preprocessing for reducing model dependence in parametric causal inference. *Political Analysis*, 15(3):199–236, 2007.
- [96] D. E. Ho, K. Imai, G. King, and E. A. Stuart. MatchIt: Nonparametric preprocessing for parametric causal inference. *Journal of Statistical Software*, 42(8):1–28, 2011.
- [97] R. L. Hopcroft. Is gender still a status characteristic? *Current Research in Social Psychology*, 7(20):339–346, 2002.
- [98] I. Hossain. Filter bubbles are shrinking our minds. The Huffington Post, November 2016. <https://www.wired.com/2016/11/filter-bubble-destroying-democracy/>.
- [99] M. Huffman. Finding the best price on amazon. Consumer Affairs, 2016. <http://bit.ly/2pTcTy0>.
- [100] How do you rank search results? Indeed, 2016. <http://support.indeed.com/hc/en-us/articles/204488980-How-do-you-rank-search-results->.
- [101] Indeed hits record 200 million unique visitors. Indeed, 2016. <http://blog.indeed.com/2016/02/08/indeed-200-million-unique-visitors/>.
- [102] K. Järvelin and J. Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’00, 2000.

BIBLIOGRAPHY

- [103] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, Oct. 2002.
- [104] T. Joachims, A. Swaminathan, and T. Schnabel. Unbiased learning-to-rank with biased feedback. 2017.
- [105] F. Karimi, C. Wagner, F. Lemmerich, M. Jadidi, and M. Strohmaier. Inferring gender from names on the web: A comparative evaluation of gender detection methods. *WWW '16 Companion*, 2016.
- [106] D. Kedmey. This is how uber’s ‘surge pricing’ works. *Time*, Dec. 2014. <http://time.com/3633469/uber-surge-pricing/>.
- [107] P. T. Kim. Data-driven discrimination at work. *William & Mary Law Review*, 58, 2017.
- [108] S. KIM. Scientists claim to have cracked how to beat uber’s surge pricing algorithm. *ABC News*, 2015. <http://abcn.ws/1Q05sgZ>.
- [109] J. Kirkpatrick. Women, work, and the state of wage inequality. *HIRED*, April 2017. <https://hired.com/gender-wage-gap-2017>.
- [110] C. Kliman-Silver, A. Hannak, D. Lazer, C. Wilson, and A. Mislove. Location, location, location: The impact of geolocation on web search personalization. In *Proceedings of the 2015 ACM Conference on Internet Measurement*, 2015.
- [111] D. Kravets. Judge calls uber algorithm “genius,” green-lights surge-pricing lawsuit. *Ars Technica*, April 2016. <http://bit.ly/2oYHWrF>.
- [112] T. Kricheli-Katz and T. Regev. How many cents on the dollar? women and men in product markets. *Science Advances*, 2(2), 2016.
- [113] P. Kuhn and K. Shen. Gender discrimination in job ads: Evidence from china. *Quarterly Journal of Economics*, 128(1):287–336, 2012.
- [114] J. Kulshrestha, M. Eslami, J. Messias, M. B. Zafar, S. Ghosh, K. P. Gummadi, and K. Karahalios. Quantifying search bias: Investigating sources of bias for political searches in social media. 2017.
- [115] J. LEBER. Algorithmic pricing is creating an arms race on amazon’s marketplace. *Fast Company*, 2016. <http://bit.ly/2pSzVHM>.

BIBLIOGRAPHY

- [116] M. Lécuyer, G. Ducoffe, F. Lan, A. Papancea, T. Petsios, R. Spahn, A. Chaintreau, and R. Geambasu. Xray: Enhancing the web’s transparency with differential correlation. In *Proceedings of the 23rd USENIX Conference on Security Symposium, SEC’14*, 2014.
- [117] E. Levenson. This map shows the boston boundaries of uber’s surge pricing. the Boston Globe, 2015. <http://bit.ly/2pi2hHr>.
- [118] J. Li, M. Ott, and C. Cardie. Identifying manipulated offerings on review portals. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP ’13*, 2015.
- [119] H. Lin. Doj intake and charging policy for computer crime matters. LAWFARE, 2016. <https://www.lawfareblog.com/doj-intake-and-charging-policy-computer-crime-matters>.
- [120] M. Lindner. Global e-commerce set to grow 25% in 2015. Internet Retailer, July 2015. <https://www.internetretailer.com/2015/07/29/global-e-commerce-set-grow-25-2015>.
- [121] Madigan probes national job search sites over potential age discrimination. Illinois Attorney General Press Release, March 2017. http://www.illinoisattorneygeneral.gov/pressroom/2017_03/20170302.html.
- [122] W. Liu and D. Ruths. What’s in a name? using first names as features for gender inference in twitter. In *AAAI Spring Symposium Series*, 2013.
- [123] S. LOHR. Banking start-ups adopt new tools for lending. The New York Times, 2015. <http://www.nytimes.com/2015/01/19/technology/banking-start-ups-adopt-new-tools-for-lending.html>.
- [124] L. Lu, R. Perdisci, and W. Lee. SURF: Detecting and Measuring Search Poisoning. 2011.
- [125] V. Luckerson. Here’s how you can avoid uber surge pricing. Fortune, 2015. <http://fortune.com/2015/10/29/tips-uber-surge-pricing/>.
- [126] A. Madhani. White house raises concerns about data discrimination. USA Today, May 2014. <http://www.usatoday.com/story/news/nation/2014/05/01/white-house-big-data-discrimination/8566493/>.

BIBLIOGRAPHY

- [127] B. Masiello. Peeking under the hood at uber. Medium, 2016. <http://bit.ly/2pmwqWO>.
- [128] E. Mazza. Uber raises fares during sydney hostage crisis, then offers free rides. The Huffington Post, Dec. 2014. <http://huff.to/18W6ybk>.
- [129] J. Mikians, L. Gyarmati, V. Erramilli, and N. Laoutaris. Detecting price and search discrimination on the internet. In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks, HotNets-XI*, 2012.
- [130] J. Mikians, L. Gyarmati, V. Erramilli, and N. Laoutaris. Crowd-assisted search for price discrimination in e-commerce: First results. In *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies, CoNEXT '13*, 2013.
- [131] T. Minkus and K. W. Ross. I know what you're buying: Privacy breaches on ebay. In *Privacy Enhancing Technologies, PETS '14*, 2014.
- [132] A. Monroy-Hernández. Nyc taxi trips. GitHub, Nov. 2014. <http://www.andresmh.com/nyctaxitrips/>.
- [133] Monster about us. Monster, 2016. <http://www.monster.com/about/>.
- [134] Monster power resume search. Monster, 2016. https://media.newjobs.com/cms/static-content/info/PRODUCTS/Monster_Power_Resume_Search_ProductSheet.pdf?intcid=PRS-DL-PHPP-20FR.
- [135] Power resume search. Monster, 2016. <http://hiring.monster.com/recruitment/Resume-Search-Database.aspx>.
- [136] C. A. Moss-Racusin, J. F. Dovidio, V. L. Brescoll, M. J. Graham, and J. Handelsman. Science faculty's subtle gender biases favor male students. *Proceedings of the National Academy of Sciences*, 109(41):16474–16479, 2012.
- [137] A. Mukherjee, A. Kumar, B. Liu, J. Wang, M. Hsu, M. Castellanos, and R. Ghosh. Spotting opinion spammers using behavioral footprints. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '13*, 2013.
- [138] V. Narayanan and A. Raman. Aligning incentives in supply chains. *Harvard business review*, 82(11):94–102, 2004.

BIBLIOGRAPHY

- [139] J. Nash. The Bargaining Problem. *Econometrica*, 18(2):155–162, 1950.
- [140] H. Nissenbaum. *Privacy in Context: Technology, Policy, and the Integrity of Social Life*. Stanford University Press, Stanford, CA, USA, 2009.
- [141] K. Novak and T. Kalanick. System and method for dynamically adjusting prices for services. US Patent App. 13/828,481, Sept. 2013.
- [142] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly. Detecting spam web pages through content analysis. 2006.
- [143] D. OBERHAUS. You’re paying too much on amazon, study says. Motherboard, 2016. <http://bit.ly/2pTfpoh>.
- [144] I. Oh. Uber will stop charging ridiculous prices during emergencies. Huffington Post, July 2014. <http://huff.to/1qJLs5c>.
- [145] W. Oremus. Who controls your facebook feed. Slate, January 2016. http://www.slate.com/articles/technology/cover_story/2016/01/how_facebook_s_news_feed_algorithm_works.html.
- [146] K. Osborn. Uber’s surge pricing doesn’t work the way it’s supposed to, says report. Money, 2015. <http://time.com/money/4092613/uber-surge-pricing-study/>.
- [147] M. Ott, C. Cardie, and J. Hancock. Estimating the prevalence of deception in online review communities. In *Proceedings of the 21st International Conference on World Wide Web, WWW ’12*, 2012.
- [148] D. Pager. *Marked: Race, crime, and finding work in an era of mass incarceration*. University of Chicago Press, 2008.
- [149] D. Pager and H. Shepherd. The sociology of discrimination: Racial discrimination in employment, housing, credit, and consumer markets. *Annual review of sociology*, 34:181, 2008.
- [150] E. Pariser. *The Filter Bubble: What the Internet Is Hiding from You*. Penguin Group , The, 2011.
- [151] J. Pitkow, H. Schütze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, and T. Breuel. Personalized search. *Commun. ACM*, Sept. 2002.

BIBLIOGRAPHY

- [152] B. Popper. Uber Kept New Drivers Off The Road To Encourage Surge Pricing And Increase Fares. The Verge, February 2014. <http://bit.ly/1hoPgU8>.
- [153] S. Preibusch, T. Peetz, A. Gunes, and B. Berendt. Purchase details leaked to paypal (short paper). In *Financial Cryptography*, 2015.
- [154] S. Pudney and M. A. Shields. Gender and racial discrimination in pay and promotion for nhs nurses. *Oxford Bulletin of Economics and Statistics*, 62(s1):801–835, 2000.
- [155] E. Rader and R. Gray. Understanding user beliefs about algorithmic curation in the facebook news feed. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI ’15, 2015.
- [156] J. Ren, A. Rao, M. Lindorfer, A. Legout, and D. Choffnes. Recon: Revealing and controlling pii leaks in mobile network traffic. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys ’16. ACM, 2016.
- [157] P. A. Riach and J. Rich. Measuring discrimination by direct experimental methods: seeking gunsmoke. *Journal of Post Keynesian Economics*, 14(2):143–150, 1991.
- [158] M. Richardson. Predicting clicks: Estimating the click-through rate for new ads. In *Proc. of WWW*, 2007.
- [159] C. L. Ridgeway. Framed by gender - how gender inequality persists in the modern world. *Oxford University Press*, 2011.
- [160] F. Roesner, T. Kohno, and D. Wetherall. Detecting and defending against third-party tracking on the web. In *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, 2012.
- [161] A. Rosenblat. Uber’s phantom cabs. Motherboard, July 2015. <http://motherboard.vice.com/read/ubers-phantom-cabs>.
- [162] C. Sandvig, K. Hamilton, K. Karahalios, and C. Langbort. Auditing algorithms: Research methods for detecting discrimination on internet platforms. In *Proceedings of “Data and Discrimination: Converting Critical Concerns into Productive Inquiry”, a preconference at the 64th Annual Meeting of the International Communication Association*, 2014.
- [163] T. Schmidt. Surgeprotector. iTunes App Store, 2014. <http://apple.co/1GxhDMU>.

BIBLIOGRAPHY

- [164] R. M. Sellers and J. N. Shelton. The role of racial identity in perceived racial discrimination. *Journal of personality and social psychology*, 84(5):1079, 2003.
- [165] A. Selyukh. Uber surge price? research says walk a few blocks, wait a few minutes. NPR, 2015. <http://n.pr/1PWniS5>.
- [166] A. Shi. Amazon prime members now outnumber non-prime customers. *Fortune*, July 2016. <http://fortune.com/2016/07/11/amazon-prime-customers/>.
- [167] N. SINGER. Your online attention, bought in an instant. *The New York Times*, 2012. <http://www.nytimes.com/2012/11/18/technology/your-online-attention-bought-in-an-instant-by-advertisers.html>.
- [168] J. L. Skeem and C. T. Lowenkamp. Risk, race, & recidivism: Predictive bias and disparate impact, 2016. <https://ssrn.com/abstract=2687339>.
- [169] G. Soeller, K. Karahalios, C. Sandvig, and C. Wilson. MapWatch: Detecting and Monitoring International Border Personalization on Online Maps. In *Proceedings of the 25th International World Wide Web Conference*, May 2016.
- [170] S. Soper. More than 50% of shoppers turn first to amazon in product search. *Bloomberg*, September 2016. <https://bloom.bg/2e5CD1o>.
- [171] Baby names from social security card applications-national level data. *data.gov*, 2016. <http://bit.ly/2f6Bz3o>.
- [172] L. B. Statistics and L. Breiman. Random forests. In *Machine Learning*, 2001.
- [173] C. M. Steele and J. Aronson. Stereotype threat and the intellectual test performance of african americans. *Journal of Personality and Social Psychology*, 69(5):797–811, 1995.
- [174] B. Stone. Uber is winning over americans’ expense accounts. *Bloomberg*, Apr. 2015. <http://bloom.bg/1IFZ0p7>.
- [175] J. D. Sutter. Amazon seller lists book at \$23,698,655.93 – plus shipping. *CNN*, April 2011. <http://www.cnn.com/2011/TECH/web/04/25/amazon.price.algorithm/>.

BIBLIOGRAPHY

- [176] L. Sweeney. Discrimination in online ad delivery. 2014.
- [177] D. K. Taft. Amazon buy box: The internet’s \$80 billion sales button. eWeek, October 2014. <http://www.eweek.com/enterprise-apps/slideshows/amazon-buy-box-the-internets-80-billion-sales-button.html>.
- [178] C. Tang, K. W. Ross, N. Saxena, and R. Chen. What’s in a name: A study of names, gender inference, and gender behavior in facebook. In *DASFAA Workshops*, 2011.
- [179] J. Thebault-Spieker, L. G. Terveen, and B. Hecht. Avoiding the south side and the suburbs: The geography of mobile crowdsourcing markets. 2015.
- [180] J. Thebault-Spieker, L. G. Terveen, and B. Hecht. Avoiding the south side and the suburbs: The geography of mobile crowdsourcing markets. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work, CSCW ’15*. ACM, 2015.
- [181] D. Tomaskovic-Devey. *Gender & racial inequality at work: The sources and consequences of job segregation*. Number 27. Cornell University Press, 1993.
- [182] M. A. Turner, M. Fix, and R. J. Struyk. *Opportunities denied, opportunities diminished: Racial discrimination in hiring*. The Urban Insitute, 1991.
- [183] L. Vaas. Uber “showing drivers and riders different fare estimates”, says lawsuit. naked security, 2017. <http://bit.ly/2pbX0VC>.
- [184] D. G. Wagner and J. Berger. Gender and interpersonal task behaviors: Status expectation accounts. *Sociological Perspectives*, 40(1):1–32, 1997.
- [185] K. WAGSTAFF. Want to avoid uber surge pricing on halloween? this study has tips. NBC News, 2015. <http://nbcnews.to/1MxAm0H>.
- [186] D. Y. Wang, M. Der, M. Karami, L. Saul, D. McCoy, S. Savage, and G. M. Voelker. Search + seizure: The effectiveness of interventions on seo campaigns. 2014.
- [187] D. Y. Wang, S. Savage, and G. M. Voelker. Juice: A longitudinal study of an seo botnet. 2013.
- [188] D. Weichselbaumer. Sexual orientation discrimination in hiring. *Labour Economics*, 10(6):629–642, 2003.

BIBLIOGRAPHY

- [189] M. Wilson. Uber fudges the position of local drivers, but they've got a pretty good reason why. Fast Company, July 2015. <http://bit.ly/2oiRb7V>.
- [190] H. Xu, D. Liu, H. Wang, and A. Stavrou. E-commerce reputation manipulation: The emergence of reputation-escalation-as-a-service. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, 2015.
- [191] S. Yuan, J. Wang, and X. Zhao. Real-time bidding for online advertising: Measurement and analysis. In *Proceedings of the Seventh International Workshop on Data Mining for Online Advertising*, ADKDD '13, 2013.
- [192] W. Zhang, S. Yuan, and J. Wang. Optimal real-time bidding for display advertising. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, 2014.
- [193] F. Zhu and Q. Liu. Competing with complementors: An empirical look at amazon. com. 2016.
- [194] P. Ziobro. Target expands online price-match policy to include amazon, wal-mart. The Wall Street Journal, September 2015. <http://on.wsj.com/2oDCMn4>.