UNIVERSITY OF CINCINNATI

Date: June 3, 2008

I, Andrew David Gilmore, hereby submit this work as part of the requirements for the degree of:

Master of Science

in:

Computer Science

It is entitled:

An Exploratory Study of the Airline Ticket Purchasing Problem

This work and its defense approved by:

Chair: Dr. Lawrence Mazlack
       Dr. Carla Purdy
       Dr. Karen Davis
An Exploratory Study of the
Airline Ticket Purchasing Problem

A thesis submitted to the
Graduate School
of the University of Cincinnati
In partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE
COMPUTER SCIENCE

In the Department of Computer Science
of the College of Engineering

By
ANDREW DAVID GILMORE
B. Sci, Computer Engineering
University of Cincinnati, June, 2003

Committee Chair
Dr. Lawrence J. Mazlack

Committee Members
Dr. Carla Purdy
Dr. Karen Davis

Date:
June 3, 2008
Abstract

This thesis is an exploratory study on airline ticket purchasing. Actual data are collected and studied with the goal of using a technique to purchase the lowest cost airline ticket given the information available. These data are studied in two ways: an algorithm based on the solution to the Secretary is applied and the trends of the data are reviewed. The Secretary Problem is where an employer is interviewing candidates for a secretarial position and the response to hire or reject a candidate must be made immediately after the interview. A sensible strategy to select the best candidate would be to interview and reject candidates without consideration to get an idea of the pool, and then at some point begin searching for the best candidate based on the initial pool. Generally, the first candidate found better than the initial pool is accepted. Buying an airline ticket can be in a way thought of as analogous to this: at each candidate ticket, one must decide to accept and purchase the ticket or reject the current ticket price and wait for a potentially better ticket price. From the data, one can see that sudden drops in price make airfare purchasing a good candidate for the Secretary Problem.

The application of a modified solution of the Secretary Problem to include heuristics noted by industry experts is successful at saving money over simply buying the first ticket offered to a given customer in about half of the collected routes, as the other cases exhibit a continuously increasing behavior. In the cases it is successful, the ticket is purchased about 1.5 to 2 weeks before departure, during a price valley. In summary, the modified Secretary Algorithm is successful as it saves money for some of the customers.
# Table of Contents

List of Figures ....................................................................................................................................... x

List of Tables .......................................................................................................................................... xii

1 Introduction .............................................................................................................................................. 1

1.1 Research Questions .......................................................................................................................... 3

1.1.1 The Secretary Problem Study ...................................................................................................... 3

1.1.2 Data Trends Study .......................................................................................................................... 4

2 Background .............................................................................................................................................. 9

2.1 Airline Industry Review ....................................................................................................................... 9

2.1.1 Airline Industry Introduction ......................................................................................................... 9

2.1.1.1 Overview .................................................................................................................................. 9

2.1.1.2 Computer Reservation Systems .............................................................................................. 11

2.1.2 Ticket Buying Suggestions ........................................................................................................... 15

2.1.3 Booking Online Travel – Shopping Agents .................................................................................... 20

2.1.4 New Generation of Airfare Shopping ............................................................................................. 28

2.1.5 Summary ....................................................................................................................................... 31

2.2 Airline Analytical Perspective ........................................................................................................... 34

2.2.1 Overview ....................................................................................................................................... 34

2.2.2 Price Prediction Methods Employed In Practice ......................................................................... 39

2.2.3 Hamlet .......................................................................................................................................... 39

2.2.3.1 Rule Learning ............................................................................................................................. 40

2.2.3.1.1 Rule Learning Overview ....................................................................................................... 40

2.2.3.1.2 Reduced Error Pruning ......................................................................................................... 41
2.5.1.6 Opponent ........................................................................................................... 60
2.5.1.7 Full Information Search .................................................................................. 61
2.5.2 Relevant ............................................................................................................... 61
  2.5.2.1 Some Probability of Not Being Available ...................................................... 61
  2.5.2.2 Two or More Competing Decision Makers ................................................... 63
  2.5.2.3 Minimizing the Expected Rank ....................................................................... 63
  2.5.2.4 Observation Cost ............................................................................................ 64
  2.5.2.5 Discounted Gain ............................................................................................... 65
  2.5.2.5 Backward Induction ......................................................................................... 66

2.6 Data Analysis Techniques ....................................................................................... 69
  2.6.1 Regression .......................................................................................................... 70
    2.6.1.1 Linear Regression ........................................................................................... 70
    2.6.1.2 Linear Regression Algorithm ......................................................................... 71
    2.6.1.3 Logistic Regression ......................................................................................... 71
  2.6.2 Neural Networks ................................................................................................. 73
    2.6.2.1 Neural Networks Overview ............................................................................ 73
    2.6.2.2 Neural Networks Algorithm .......................................................................... 77
  2.6.3.1 Decision Trees Overview ................................................................................. 78
  2.6.3.2 Decision Tree Induction .................................................................................... 79
    2.6.3.2.1 Gini Index ..................................................................................................... 80
    2.6.3.2.2 Gini Diversity Index ..................................................................................... 81
    2.6.3.2.3 Information Gain .......................................................................................... 81
    2.6.3.2.4 Twoing Rule ................................................................................................. 82
2.6.3.3 Optimal Termination ................................................................. 82
2.6.3.4 Pruning ................................................................................. 84
   2.6.3.4.1 Cost-Complexity Pruning .............................................. 84
   2.6.3.4.2 Reduced Error Pruning ............................................... 86
2.6.3.5 Decision Trees Algorithm .................................................. 86

2.7 Comparison of Techniques ....................................................... 87
   2.7.1 Decision Trees ............................................................... 88
   2.7.2 Neural Networks .............................................................. 88
   2.7.3 Rule learning ................................................................. 89
   2.7.4 Regression .................................................................. 89
   2.7.5 Q-learning ................................................................. 90
   2.7.6 Moving Average ............................................................ 90
   2.7.7 Stacked Generalization ................................................... 90
   2.7.8 Solution to Secretary Problem ......................................... 90
   2.7.9 Algorithm Tradeoffs ..................................................... 91
   2.7.10 Algorithm Selection ................................................... 91

3 Technology .................................................................................. 93
   3.1 Introduction ................................................................... 93
   3.2 Protocols ........................................................................... 94
   3.3 Compression ................................................................... 97

4 Approach ................................................................................... 98
   4.1 Experimental Design ......................................................... 98
   4.2 Data & Results .................................................................. 101
7.6.4 Negative Jumps........................................................................................................... 274
7.7 Lead Time Study.............................................................................................................. 293
7.8 Data Extended................................................................................................................ 312
List of Figures

Figure 1: Farecast following 30-day cost graph for BOS to SFO to determine time to travel

Figure 2: Flyspy following 30-day cost graph for BOS to SFO to determine time to travel

Figure 3: Farecast three-month history example (www.farecast.com)

Figure 4: Kayak three-month history example (www.kayak.com)

Figure 5: FareCompare one-year history example (www.farecompare.com)

Figure 6: Hotwire’s TripStarter history example (www.tripstarter.com)

Figure 7: Farecast prediction example (www.farecast.com)

Figure 8: Availability Dynamics Example for different fare classes

Figure 9: K is somewhere between M and N

Figure 10: Sample values for M and N

Figure 11: L(w|n=10,y=7) graph

Figure 12: Decision Tree Example: decide whether a person should be offered a loan

Figure 13: Secretary Algorithm on BOS-SFO route combined over each website

Figure 14: Annotated example for Secretary Algorithm with a single customer

Figure 15: Successful Secretary Algorithm on DAY-SFO route from Continental

Figure 16: Moderately successful secretary algorithm on SEA-IAD route from Delta

Figure 17: Algorithm in action on DAY-SFO on Delta

Figure 18: Lead time study example – three weeks and longer lead time is beneficial

Figure 19: Lead time study example – one to five weeks out has varying levels of success

Figure 20: Original algorithm results (light blue) and adjusted algorithm results (yellow) example
Figure 21: Original algorithm results (light blue) and adjusted algorithm results (yellow) example

Figure 22: Original algorithm results (light blue) and adjusted algorithm results (yellow) example

Figure 23: Trend example with more activity in the middle

Figure 24: Trend example with an increase in activity as the departure date approaches

Figure 25: Decrease in activity as the departure date approaches

Figure 26: Trend example showing increased activity at boundaries: one week, three week, and four week

Figure 27: Seven-day boundary graph for every website over the route BOS to SFO

Figure 28: Trend example with more activity under the seven-day boundary
List of Tables

Table 1: Customer must begin searching for a ticket the above number of days before plane departure for the Secretary Algorithm to save money

Table 2: Number of customers (and percentage) which saved money with the Secretary Algorithm

Table 3: Average savings per customer when savings are possible

Table 4: Average loss per customer when using the theoretical Secretary Algorithm

Table 5: Average savings and losses per customer for the adapted algorithm

Table 6: Savings and losses, dollar amounts

Table 7: Buckets for BOS-SFO on Expedia

Table 8: Average number of times the price changes per day

Table 9: Minimum prices possible for each airline and route pair

Table 10: Average prices for each airline and route pair

Table 11: Breakout of airlines tickets potentially purchased from Expedia

Table 12: Breakout of airlines tickets potentially purchased from Expedia including those at the same price

Table 13: Ticket cost of algorithm ticket price minus the initial ticket price for the first customer

Table 14: Ticket cost of algorithm ticket price minus the initial ticket price average

Table 15: Ticket cost of algorithm ticket price minus the initial ticket price average after data replacement

Table 16: Average ticket purchase prices per customer per route and subsequent calculations
1 Introduction

How about a trip? Maybe a vacation to a warm sunny beach, with a beautiful blue sky, white sand, and a clear ocean? Or how about a trip to see family or what about taking family on a trip! Perhaps to see the Eiffel Tower and the Arc de Triomphe or visit the Great Wall of China? The aim is to make the trip, to where ever it might be, a little more affordable.

The question: when is the best time to buy an airline ticket? Is it one month before departure? One week? Many factors influence ticket prices such as demand for a particular flight, seasonal changes and holidays, ticket availability, and estimated number of seats left on a given future date. Quantifiable numbers for most of these influences are generally unavailable, so one must observe the available information and infer predictions. On the World Wide Web prices from airlines are available to anyone, anytime. It is commonly known, these prices fluctuate over time as the trip approaches. The price changes several times a day. Unfortunately, demand for tickets is generally unknown. Holidays are an exception where demand is known to be higher ahead of time. In particular, the number of seats left on a flight is not known. For any given flight, tickets are sold by the airlines themselves as well as resellers, so even knowledge of the number of tickets left at any one seller does not represent knowledge of the entire number of seats remaining available on the airplane. These factors lead to this exploratory study to delve into some interesting characteristics of airline ticket pricing.
When looking for an airline ticket only the current offering price is available, whether the price will increase or decrease is not known; one can only record the previous prices offered. When looking at historical ticket price trends one can occasionally find sudden drops in price for a few days. Thus, an algorithm that utilizes drops in price may be useful to a consumer. One algorithm that does this is the solution of the classic problem known as the Secretary Problem, also called the Marriage Problem, which are more technically known as optimal stopping problems. The Secretary Problem is a problem where an employer is looking to hire a secretary, and he or she can only base a hiring decision on the current candidate and past candidates. It is assumed that when a candidate is rejected, he or she will find other employment and cannot be recalled. Thus, the employer can interview and reject some candidates to get an idea of the best candidate’s qualifications, and then later select a candidate that is better than the previously interviewed candidates. One could say this is similar to the process of looking for an airline ticket. One must decide whether to purchase the current ticket price being offered or wait and hope that a cheaper ticket will be offered later. This is determined by looking at the historical information for the selected route.

This paper is an exploratory study into airline ticket pricing. Airline ticket data are complex. The basic supply and demand model says that market balance occurs when supply and demand are equal. Airline ticket data appears complex because one key piece of information is kept hidden from consumers: namely the number of seats left available on a given airplane at any particular point in time. Another way to think of it is having the knowledge of when tickets are being sold for a particular flight. Without
this knowledge, it is not known whether a price increased because the airline’s model predicts it will make more revenue doing so at that particular time, more tickets were sold than predicted and the price is increased to recover revenue, or the price changed for some other reason. Thus, the seemingly unpredictable nature of airline ticket data and the possible cost savings for airline consumers makes this a desirable area for an exploratory study. The behavior of the data is studied and an algorithm is implemented that looks for the optimal stopping point when to purchase a ticket.

1.1 Research Questions

The hypotheses approached by this thesis, the reasons they are important, and other interesting questions that may be asked after determining their conclusions follow.

1.1.1 The Secretary Problem Study

I. a) Hypothesis

The solution to the Secretary Problem can save consumers money on airline tickets.

b) Importance

Many people use airliners to visit family, travel for vacation, and their work. Saving money on airline tickets would improve the prosperity of consumers. Money saved on airline tickets can then be invested towards any other aspect of life.
c) **Follow-up**

If it is shown that money can be saved on airline tickets, further research can be put into enhancing the algorithm used here, or research effort could be put into employing other, similar algorithms.

### 1.1.2 Data Trends Study

II. a) **Hypothesis**

The prices of airline tickets monotonically increase as time progresses towards the departure date.

b) **Importance**

This is important because if this were true, the best ticket price would be a great distance from the departure date, allowing consumers to purchase a ticket early and be confident they paid a good price.

c) **Follow-up**

If this were true, the next question to ask is generally how close to the departure date does the price begin to increase so the consumer knows the latest date they can purchase a ticket before the price begins to decrease. If false, the point or points at which the monotonically increasing behavior is broken can be investigated for patterns and used to provide consumers with an insight into when to find a discount airline ticket.
III. a) **Hypothesis**

The final week before departure is the worst week to buy tickets.

b) **Importance**

Knowledge of the typical ticket prices in the final week will alert consumers to avoid the final week before departure.

c) **Follow-up**

If proved false, consumers could use the final week to their advantage. If true, a tool similar to the one covered in the other research question would enable consumers to save money.

IV. a) **Hypothesis**

The following related hypotheses are on the topic of price volatility.

i) The ticket prices change frequently. This question is confirming the literature such as McCartney’s [1997] article which discusses airlines’ use of technology to get as much money as possible from customers. The article illustrates the change in ticket prices as the departure date approaches and tickets are purchased.

ii) The price changes more often as the departure date approaches. This is to confirm the statement by Piga [2006] that fares are less stable when closer to the departure date.
iii) At the end of each seven-day interval from departure there is a sudden change in price. This question comes from the literature and speculation around the 28-day, 21-day, 14-day, and 7-day boundaries [Piga, 2006; SoYouWanna.com, 2006].

b) Importance

The knowledge of the price structure, particularly the volatility, will empower the customer to make a more informed decision on purchasing a cheap airline ticket. If the price changes a lot, the consumer, or a program on behalf of the consumer, can watch the short-term change in price and avoid a spike in the price and buy when the price is low within that small window. Hypothesis (ii) could potentially inform the consumer to be aware of an increasing amount of change. Hypothesis (iii) could potentially inform the consumer to be wary of volatility around each 7-day interval boundary.

c) Follow-up

If false, the consumer will know that they do not need to worry about short-term volatility in the price. If true, the consumer needs to be alert and watch for each of these types of volatility. This volatility knowledge can be inserted into the toolset used to search for a cheap airline ticket. The data can be further studied to predict when the prices will be most volatile. If true, the volatility could also be further studied for patterns.

V. a) Hypothesis

A larger lead time allows for greater airline ticket savings.
b) **Importance**

If true, this will give the consumer the knowledge to begin looking for a ticket early and at the appropriate time. If false, the consumer may look for a ticket at any time.

c) **Follow-up**

If true, a study could be launched to determine precisely when a consumer would benefit the most to begin to look for an airline ticket on a particular route.

VI. a) **Hypothesis**

There are a small number of unique prices for a given flight. This is following statements in the literature such as the one given by McCartney [1997] concerning the tendency for there to be few given ticket prices.

b) **Importance**

Given the buckets are similar for different days of the same flight number, the offering of a small number of unique prices would give the consumer the knowledge of the target lower priced buckets. The consumer would have the target buckets to look for and that a price between the expected buckets is not likely.

c) **Follow-up**

Knowledge of a bucket system would simplify the construction of an algorithm to search for cheap airline tickets since the classes of prices would be distinct. One could investigate a pattern for the buckets and graph bucket distributions.
The thesis is organized as follows. In chapter 2, the airline industry is discussed and the Secretary Algorithm reviewed. In chapter 3, the technology to gather the prices is discussed. In chapter 4, the approach and results are discussed. In chapter 5, the conclusions and future work are presented. Chapter 6 contains the citations. Finally, chapter 7 contains the appendices of graphs and data.
2 Background

This chapter gives an overview and detailed background into the topic. This chapter includes a review of the airline industry, the industry from an analytical perspective, covers the Secretary Problem, and compares other approaches to this problem. Those not interested in the industry background can skip the first section; 2.1.5 contains a summary. Those not interested in the technique comparisons can review only the sections specific to the Secretary Problem: 2.2 and 2.3.

2.1 Airline Industry Review

This section reviews the airline industry, the computer reservation systems used by airlines, booking online travel, and the features new websites provide. Those not interested in an overview of the industry may feel free to skip this section.

2.1.1 Airline Industry Introduction

This section contains an overview of the airline industry and the technology behind the industry.

2.1.1.1 Overview

The airline industry employs a mathematical science known as yield management to predict the optimal price for selling a ticket for the most revenue, according to McCartney [1997]. A customer could have paid $1,000 more than the one in the next seat. Prices are generally inflated at large airports [McCartney, 1997]. This is well known and
exemplified by one example where a flight from Philadelphia to Houston on Continental was $1,484 while the same trip from Baltimore, a ninety-minute drive, was $280 [McCartney, 1997]. One Wall Street Journal article [McCartney, 1997] indicates how the yield management model gouges business customers, since they are typically “more swayed by convenient schedules and frequent-flier programs than price,” while yield management offers tickets at a reduced price to leisure customers to entice them to fly. The system is able to fill flights to record capacities, to more than 70%, with many popular flights completely filled while previous systems did not do nearly as well [McCartney, 1997]. For example, “one day before departure with 130 passengers booked for a 125-passenger flight, American still offered five seats at full fare because the computer database indicated 10 passengers were likely not to show up or would take other flights” [McCartney, 1997]. The flight departed full with no one bumped [McCartney, 1997].

Ticket prices for a particular flight are varied. The person in the next seat might have paid half the price. This is confirmed by the data; ticket prices are spread out. In one example, 125 seats on a flight from Chicago to Phoenix are divided among seven fare buckets with round-trip tickets ranging from $238 to $1,404 [McCartney, 1997]. A fare bucket is a particular price, or small price range, for which a set of tickets are sold. The offering price for a traveler is dynamically chosen from a particular bucket based on factors such as customer status (such as leisure, business, or frequent flyer), number of seats available, historical patterns, and connecting passengers likely to use the route as one leg of a larger trip. The Wall Street Journal article [McCartney, 1997] gives an
example where 69 of 125 coach seats are already sold four weeks before a departure, American’s algorithm began to limit the number of seats in lower priced bucket. A week later, it totally shuts off sales for the bottom three buckets. There are also flukes in the leg pricing. On a New York to Phoenix flight with a stop in Chicago, a Chicago customer flying to Phoenix sees a higher price because the overall fare structure for that one-stop trip was higher and a lot of seats were available on the connecting New York-Chicago flight [McCartney, 1997].

2.1.1.2 Computer Reservation Systems

This section reviews the computer reservation systems used by the airline travel industry. Travel search websites, generally, are only front-ends to massive databases and mainframes, some of which have been in existence since the 1960s. These systems are called computer reservation systems (CRS); the larger ones are also known as global distribution systems (GDS). These two terms are generally synonymous. These systems store vast amounts airline information: routes, legs, times and dates, jets, and, of course, prices. Prices are updated many times per day, following supply, demand, and competition. There are four major GDSs: Amadeus, Galileo, Sabre, and Worldspan [Das, 2002]. A report chaired by Cunningham [2006] states, “Sabre, established by American, is now a publicly owned company. Apollo, also known as Galileo International, was founded by United and is now owned by Cendant Corporation. Delta (40%), Northwest (34%) and American (26%) own Worldspan. Air France, Lufthansa and Iberia effectively control Amadeus, which is headquartered in Spain. It is moving towards becoming a public
company” [Cunningham, 2006].

Additionally, there is an organization called SITA (Société Internationale de Télécommunications Aéronautiques) that serves the telecommunication needs of the international airline sector [Kärcher, 1996].

GDSs are separate entities; many different sources access them such as airlines, consolidators, and travel agencies. To ensure the correct data gets from the correct airline to the correct GDSs with the correct rules, a group called the Airline Tariff Publishing Company (ATPCO) gathers the data and updates the GDSs. According to FareCompare’s CEO Rick Seaney [2006]:

“ATPCO was originally a government agency until deregulation in the late 1970’s. It is now owned by several dozen airlines and acts as the distribution hub for over 60 million air fare filed worldwide. Each airline files their air fares continuously during the day to ATPCO. …For U.S/Canadian fares distributions occurs on weekdays at 10:00am, 12:30pm and 8:00pm and on weekends at 5:00pm EST. The airlines and online agencies then load these fares onto their quoting/booking systems within 2-6 hours. The evening feeds are loaded between 12:00am and 1:30am the next day” [Seaney, 2006].

Additionally, “the bulk of all flight schedules are filed by the airlines with OAG (Official Airline Guide) and Innovata. They act as the distribution hub to quoting/booking sites for flight information. …The quoting/booking sites get instant flight update notifications from the airlines each day so that last minute
issues (cancellation, mechanical failures, etc) can be posted immediately. A particular flight-information record includes the maximum seat count; cabin information as well as defining which inventory buckets can be sold for which seats on the plane.” Furthermore, “inventory is the key variable in the quest for the best price and is ultimately controlled by the airline. The airline has sophisticated systems that use historical models and current real-time sales information to decide how many seats they will sell at a particular price point on a particular flight at a particular point in time. This is called ‘yield management’ and it is a very powerful tool the airlines use to maximize profit” [Seaney, 2006].

GDSs were originally created by airlines and are owned by them. Over the years, these systems have been sold and merged and split as airlines are bought and sold. In the last several years, startup companies have emerged in attempts to gain market share from the GDSs by updating the antiquated technology and techniques they are using as well as boasting lower prices for the airlines to book tickets. ITA Software is one new major competitor. According to Devlin [2006]:

“[ITA’s] story began a few years ago, when Jeremy Wertheimer, then a graduate in computer science at MIT, and a group of fellow graduate students set out to develop an airline pricing search engine. When what looked like being a few months work failed to yield the expected results, the group began to examine the nature of the airline pricing system. Their study did lead them to develop a powerful price-searing system, which the company they formed, ITA software, markets to both Orbitz (a fare searching site owned by five of the major US
airlines) and Delta Airlines, who use it to drive their search utilities. But it also led to the discovery, by research team member de Marcken, that they were chasing the end of a rainbow” [Devlin, 2006].

Kayak also uses ITA Software’s product [Smith, 2005a].

Bonné [2006] explains: ITA uses modern distributed computing techniques to search airfare data instead of traditional massive mainframe-based systems. Another major advantage is their flight caching system. If every passenger poses 100 searches before buying a ticket and each search looks at 1,000 flights, then the airlines would need to answer 15,000,000 questions a second. Neither their networks nor their computers can handle this, a situation that forced ITA Software to develop a sophisticated seat availability caching system [de Marcken, 2006]. They also intuitively check fares for nearby airports [Bonné, 2006]. Traditional GDSs are being forced to update and expand their services to defend their positions and market share [Erickson, 2006]. A report chaired by Cunningham states [2006], “GDS collect a fee of about $4.30 on average from the airline for each segment sold” [Cunningham, 2006]. Colson [2005] is quoted, “ITA Software says it will charge airlines US$0.40 per segment” [Colson, 2005].

This section overviewed the technology the airlines use to calculate and process airline ticket cost and demand. The result of this analysis from the consumer point of view is airline ticket prices offered for desired routes on particular days and range of times. This information is stored on the GDSs. The next sections discuss the system from the consumer’s point of view. In particular, the following sections cover
suggestions to purchase cheap tickets, booking travel online, and advances in online travel agencies.

2.1.2 Ticket Buying Suggestions

This section discusses ticket buying suggestions. Shopping for an airline ticket is often undertaken using a set of guidelines based on consumer experience in an attempt to find a lower fare. Many of these tips are presented here. Demand explains why it is cheaper to fly on weekday or odd hours: there are fewer people flying than on weekend or holiday flights and the airlines want to fill airplanes [SoYouWanna.com, 2006]. More popular routes and dates are going to be priced higher and less popular routes and travel times are priced lower. Holiday travel is more expensive since many consumers are visiting with family. Saturday night stays are cheaper since airlines separate business and casual travelers; business travelers typically are not willing to stay a Saturday night and they are willing to pay a higher price to do business quickly [Elliot, 2006]. Fridays and Mondays are the most expensive times to fly and weekends are in high demand. Tuesdays and Wednesdays are the cheapest days to fly [SoYouWanna.com, 2006].

SoYouWanna.com [2006] suggests making a reservation at least 21 days before departure. The site also indicates that last-minute deals, if available, may be the cheapest since airlines want to fill a flight. At the reduced cost, the last minute deals offer little flexibility. Customer date flexibility also aids the low fare search. According to Seaney [2006], CEO of FareCompare, “there are occasions in certain city
pairs where last minute purchases can be a super deal. These are perfect for those with ultimate flexibility and who can pick up and go at a moments notice.” Airlines typically attach restrictions to discount fares, like a 7-, 14-, or 21-day advance purchase and/or a Saturday night stay. Additionally, if schedule flexibility is possible, the cheapest flights are usually in the winter, except for Thanksgiving and Christmas/New Year’s time [SoYouWanna.com, 2006]. Further, consider nearby airports. Secondary airports outside the city or in nearby cities typically have reduced prices since they are less popular [SoYouWanna.com, 2006]. Finally, when applicable, check for student and senior discounts [SoYouWanna.com, 2006].

Another place to shop is at a consolidator, an intermediary company that buys tickets at a discount directly from the airlines. Customers benefit from their rates, but customers have to be careful that the business is reputable; in recent years, some consolidators have gone out of business and left customers in the dark at the airport [SoYouWanna.com, 2006]. The Internet is often just used as a research tool. Not everyone who finds their best rate online actually buys online, instead turning to the airline or a travel agent. The Internet is handy for buying a ticket as quickly as possible from the convenience of any location [SoYouWanna.com, 2006]. Travel agents have an edge over Internet travel sites because they are trained to use the central reservation system to uncover information. They also can check fares on lesser-known airlines not in reservation systems. Consumers should be aware that travel agents are often paid by commission so they may not work hard to find a cheap ticket [SoYouWanna.com, 2006].
Another option for finding cheap airline tickets is an auction. There are generally two types: typical (English) auctions or reverse (Dutch) auctions. At a typical auction site, the seller puts an item up for sale and sells to the highest bidder. At reverse auctions, individuals specify the price they pay for a seat and the airline decides whether or not to agrees to the price. SoYouWanna.com [2006] says, “these sites are fantastic because you can choose your own price, and you might get the ticket. The drawbacks are that: 1) you have to put in your credit card number before you know the exact times of flight, so you’re stuck with whatever you get, and 2) you often have to fly at crazy hours.” Finally, there is also the option of being a courier. A courier is someone who travels without luggage so the company can use the space to send goods or papers. This method might save 50% or more, but the passenger is limited to only a carry-on [SoYouWanna.com, 2006]. Being a courier is risky since the ticket is in control of the courier company until it is handed over the day of the flight and the contact might not arrive on time [Gregory, 2006].

Booking a lower-fare flight is often an enormous task. One needs to compare prices from many airlines and travel agencies over the desired dates and times. This is complicated by the fact that the airline industry, like any industry, has the goal of maximizing its revenue. According to Seaney [2006], CEO of FareCompare, “the business and leisure price break point is normally 14 days advance purchase (sometimes 21 days). As most business travelers know – you are not likely to get a good price inside of 14 days.” One well-known travel tip is to purchase tickets Wednesday at midnight, yet this tip is refuted below. Ashley [2006] of UpgradeTravel asked Rick
Seaney, CEO of *FareCompare*, about the Wednesday at midnight rule,

“held inventory is released every day at midnight so Wednesday is nothing special. It’s not the low fare inventory that opens up at midnight. Low fare inventory is almost always ticketed immediately. Un-ticketed inventory is normally high priced business inventory held by a corporate agency for business travelers who are on the fence about going, or by government workers who have a special ‘hold until travel’ feature for negotiated routes. New fares (lower or higher) are distributed at 10:00am, 12:30pm, and 8pm EST and loaded about 2-6 hours later. Seat inventory is controlled by automated revenue management systems which continually monitor current sales and consult historical models to decide on whether to release the lowest price seat inventory. The 8pm domestic … fare feed (5pm weekends) is loaded … between 12:15am and 1:30am, which has the changed fares. But there is no correlation to getting a good deal, just because some inventory might be freed up at midnight. It is just as likely to free up at 2pm when the yield management system decides sales are soft in a particular inventory price bucket for a particular flight” [Ashley, 2006]. As illustrated later, “price buckets” are particular prices, or tight price ranges, for which groups of seats are sold. If seats are not selling as well as expected, the tickets will be moved from their current price bucket to a lower price bucket.

Livingston [2006] provides general trends about ticket price offering changes as a departure date gets closer,

“about 11 months before your travel dates is when airlines start offering tickets.
Around 90 days before departure, the airlines start to compare the number of tickets that have been sold to the number their computer models predicted. If the numbers are too low, the airlines will drop some prices until the number of filled seats is back on target. Prices start to go up after the 21-day advance purchase deadlines, and go up yet again at the 14-day and 7-day purchase deadlines. Within 6 days of a flight, the airlines know that [business] prospective travelers have little flexibility in their travel plans. Higher prices in this down-to-the-wire period are usually the result” [Livingston, 2006].

Cowen [2006], “The Internet Travel Guru,” provides additional tips as follows:

1) Orbitz and Kayak offer an email alert when a specified price drops below a specified value.

2) When changing a ticket because of a price drop, some airlines will provide a voucher for the difference upon request, possibly minus an administrative fee.

3) Occasionally multiple leg flights are priced cheaper than a particular leg of a flight. For example, buying a four-leg ticket for a two-day flight could possibly save money [Cowen, 2006].

McGee [2006] warns to double-check the amount that will be charged to a credit card. The price often rises during the check-out process [McGee, 2006].

This section covered some suggestions found in the literature. A few of them will be incorporated into later studies and some others shown to be true through analysis. The next section will discuss online travel agents and purchasing an airline ticket online.
2.1.3 Booking Online Travel – Shopping Agents

This section reviews booking airline travel tickets online. There are several online start-up companies offering powerful searches and systems to quickly provide the consumer with information that would take much time to gather through conventional airline and online travel agent searches. Over the past decade the major online travel agents – Expedia, Travelocity, and Orbitz – have been taking market share from the office and phone based travel agents causing them to close. Now, there is a similar effect happening to the traditional online travel agents and they are trying to keep up with the startups [Gannes, 2006]. Farecast and Flyspy are developing systems to show consumers the best time to fly over a thirty-day period given the start, destination, and length of trip desired [Cubrilovic, 2006]. See figures 1 and 2 for illustrations of the cost of the lowest offered airline tickets for departure dates over the following thirty days. Figure 1 shows the lowest offered ticket prices from FareCast between San Francisco and Boston from December 25 to January 23 for trips of length two to eight nights. Figure 2 shows the lowest offered ticket prices FlySpy scanned between Minneapolis and London Gatwick from December 25 to January 23 for trips of length three nights. Both of these figures show that departures near the current date are most expensive, which also happens to be a holiday. Figure 1 shows that trips departing at the beginning of the weekend – Thursday and Friday – as well as those departing at the end of the weekend – Sunday and Monday – are more expensive than the other days. Figure 2 shows different kind of information: it is looking at a subset of prices that figure 1 is – three night trips. It shows that this class of trips have basically two tiers – cheap when on a Thursday, Friday, or Saturday and extremely costly otherwise.
Currently *Flyspy* only has data available for flights leaving Minneapolis. Along
similar lines to Flyspy, Farecast originally only allowed users to choose departure from Seattle or Boston [Arrington, 2006]. Farecast now presents predictions for seventy-five cities [Kirkpatrick, 2006]. Additionally, Farecast provides a grid showing the price of flights for a two to eight day trip over the range of a specified two-week period. The only similar functionality offered on airline and online travel agencies is to be able to search plus or minus one to three days from the specified dates. Another powerful feature sites are using is the ability to search nearby airports. Since larger airports are often more expensive, having the ability to retrieve results from nearby airports without manually performing those searches saves copious amounts of time. To remain competitive, traditional online travel agents are being updated to compete with the startup companies [Erickson, 2006].

Farecast’s RSS service also provides a unique feed according to Smith [2006]: fare information for a flight on particular dates between a particular pair of cities. Other RSS feeds offer the following services: Kayak offers Fare Alerts for a maximum price, but the service looks at a full month as opposed to a particular date. FareChase’s service does not allow a specific date or date range; it only provides deals on a given flight. Orbitz provides deals for specified destinations but nothing for a particular date or city pair. Travelocity offers an alert if a price changes by a specified amount between two cities, but not for particular dates. Expedia allows the tracking of a city pair, but not a date or specific fare. FareCompare allows for the tracking of a city pair, but there is no way to specify a date [Smith, 2006].
Some sites are now providing historical data for the prices of similar tickets being sold over past months or years. This information being available serves two purposes in particular. Short-term data shows whether the price trend is increasing, decreasing, or remaining stable. When a price is increasing, it is more likely to continue increasing. Check back an hour or two later and make sure the price offered was not a spike and then purchase the ticket before the price goes any higher [Seaney, 2006]. When a price is decreasing, it is likely to continue decreasing or remain stable. It is advisable to wait until the price reaches a valley before purchasing, but do not wait too long or the price will increase when the demand surpasses the supply. Seaney [2006] recommends, “check the volatility of your city pair. If the prices are bouncing around by $100 or more a few times a month or more, make sure you buy on a down swing. Know when the best time to buy was last year. [The history] doesn’t mean it will happen this year but it is part of a ‘savvy’ air travel shopper’s arsenal” [Seaney, 2006]. This history information can be found on FareCompare and TripStarter for select cities and on some routes for Farecast and Kayak; see figures 3, 4, 5, and 6 for history examples. Figure 3 shows a graph of the lowest airline ticket prices from FareCast for the given route over the past ninety days, the trip here being December 27 to January 3 from Boston to San Francisco, and also shows the lowest price over the period and the average price during the period. This graph is interesting because it shows a trend of slightly increasing prices as the holidays approach with a sudden drop two weeks before Christmas and then a large increase for the holiday. Figure 4 shows an example of the Best Fare Trend graph of the lowest cost airline tickets offered by Kayak for the past three months for the same route. Similar to Farecast, this graph shows a small increase as the holidays
approach, a large increase two weeks before Christmas, and a large increase for the holiday. Figure 5 shows the airline ticket price trend – the minimum, average, and maximum of the lowest ticket prices for each month – for the past year over the given route, Boston to San Francisco in this case, from FareCompare. This graph allows the savvy shopper to see the trend for the same month a year earlier to get an idea if the currently offered price is reasonable. This allows a shopper to review expected prices that a ninety-day graph cannot as seasonal and holiday demand are not revealed in short-term history graphs. Figure 6 shows the average price of airline tickets for the previous year as well as the current year to date for the route Boston to San Francisco.

TripStarter shows more detail than FareCompare’s one number per month. These two graphs show an increase in price for Thanksgiving, Christmas, and early summer. TripStarter goes one step further than FareCompare and provides the customer with the suggested cheapest months and best holidays to travel for the given search.

Fare History for SFO > EOS (Dec 27 - Jan 3)

Figure 3: Farecast three-month Daily Low Fare History example (www.farecast.com) [Image Screen Capture, December 24, 2006]
Figure 4: Kayak three-month Best Fare Trend history example (www.kayak.com)
[Image Screen Capture, December 24, 2006]

Figure 5: FareCompare one-year history example (www.farecompare.com)
[Image Screen Capture, December 24, 2006]
There are generally four types of airfare search websites: airlines such as Delta and Continental; online travel agencies such as Expedia, Orbitz, and Travelocity; meta sites such as Kayak, Sidestep, Mobissimo, and Yahoo’s Farechase; and auctions such as Priceline. Some consider auction sites to be a type of online travel agency that sells tickets in a different manner. Online travel agencies purchase tickets in bulk from airlines and resell them to travelers. Meta sites search airlines and online travel agencies for ticket prices and schedules and then transfers customers directly to those sites for booking. The meta sites receive money when a potential customer clicks over to the airline or online travel agency [Rand, 2006]. Meta sites believe their offer is better because they search over 100 websites to save the customer time. Additionally, meta sites search some low-cost carriers, such as JetBlue, which generally do not participate with the online travel agents [Rand, 2006]. Mobissimo claims to more intelligently search alternate routes when major hubs are sold out [Smith, 2005b]. Additionally,
there are a few other types of sites that exist for airline travelers. There are social websites for users to discuss their plans or experiences and post pictures. *TripHub* and *RealTravel* are examples [Fehrenbacher, 2006]. Finally, another interesting website is *AirfareWatchdog*. It is a website that is dedicated to finding unpublished deep-discount fares [Bly, 2007].

Low-cost carriers is an interesting subclass of the major airlines. These airlines, more recently launched than the major airlines, have business models specifically to price tickets cheaper than the major airlines. The two major competitors in the U.S. are Southwest and JetBlue. These low-cost carriers save money by offering the bare minimum of amenities [Ellis, 2006]. Interestingly, Southwest does not allow most websites to use their price and schedule data and to redistribute tickets [Pallatto, 2001]. They do have an agreement with *Sabre*, the owner of *Travelocity*, for ticket distribution [Business Wire, 2005]. In addition to online travel sites, travel agents have affiliate agreements with large travel companies and negotiate lower rates, although travel agents are not obligated to find the best possible deal and they are often paid commission [Elliot, 2006].

This section presented sample features and examples from newer growing websites. The figures show interesting characteristics, some of which are seen later. The next section discusses a new online airfare shopping technology, the precursor for this research.
2.1.4 New Generation of Airfare Shopping

This section covers a new feature to ease airfare shopping. *Farecast* offers a unique and particularly interesting service: airfare predictions. Given a particular route and dates *Farecast* will generate a prediction of whether the price will sharply increase, mildly increase, stay stable, mildly decrease, or sharply decrease over the following seven days as well as the confidence in that prediction and a recommendation of buying or waiting for a better price. *Farecast* reports statements such as: there is an 80% certainty that the lowest fares will rise by $50 over the next seven days [Arrington, 2006]. See figure 7 for an example showing a prediction tip of buying now with the lowest fares holding steady or rising, but with a confidence under 50%. Studies have shown that the prediction is roughly seventy to seventy-five percent accurate [Kirkpatrick, 2006]. Currently this feature is only supported on a limited number of airports and dates. To defend the prediction, a service called Fare Guard guarantees the price. According to Kirkpatrick [2006], “if *Farecast* tells you a ticket’s price is going to drop and recommends that you wait, you can pay the Fare Guard fee to lock in access to the lowest price of that day for the next week. If you have purchased Fare Guard and the price instead goes up, *Farecast* will send you the difference between what you ended up having to pay and the price you locked in with them. If *Farecast’s* prediction was correct and the price does drop, you can buy the ticket at the lower price.” Metzbaugh [2006] points out that Fare Guard is only offered on tickets it predicts are going to fall or hold steady over the next seven days. Furthermore, Marshall [2006] says that *Farecast* only looks out 90 days into the future and only predicts fares for trips of two-to-eights days in length.
Cook [2004] quotes Oren Etzioni, Farecast’s founder and a University of Washington computer science professor saying,

“[airfare] is quite predictable. Critics say predicting airline ticket prices is impossible, arguing that they rise and fall based on dozens of variables. Others make the point that airlines could simply reconfigure their pricing system to override [such] technologies. Undermining the pricing of airlines is not [the] goal, with Etzioni saying he doesn’t view the relationship with the carriers ‘as an antagonistic situation.’ He compares [to] the startup of [the] Priceline.com company that works directly with airlines to sell unfilled seats at a discount. America West Airlines spokesman Carlo Bertolini doesn’t see the technology as a threat in an industry where airline prices have become increasingly transparent with the rise of comparison shopping on the Internet. In fact, he wonders how the airlines could use the technology to cut their own costs” [Cook, 2004].

Farecast also has a marketing relationship with American Airlines, making it easier to refer directly to AA’s website as well as giving Farecast access to AA’s lowest fares [Hartzell, 2006]. The functionality provided by Farecast is endorsed by American Airlines. The article by Hartzell [2006] states: “[American Airlines supports] Farecast
because they provide useful information that helps consumers make smarter and more confident purchase decisions,’ said David Cush, American Airlines senior vice president, Global Sales. ‘Farecast’s model supports direct AA.com booking, where travelers can benefit from AA.com’s lowest fare guarantee.’” Peterson [2006] points out that Farecast gets its data from ITA Software and Farecast says it now has 50 billion historical price quotes from routes to study. Ms. Peterson also says that no other company is in the business of airfare prediction. “Farecast is trying to help the consumer answer a fundamentally different question, which is when to buy, versus the question of what to buy” [Peterson, 2006].

Farecast uses data mining on airfare pricing data to aid customers in determining when to purchase airline tickets. Data mining is also being used on airline mileage program card holders. It is being used to give customers incentives to fly with that airline, and it is generating customer loyalty by providing rewards programs [McKnight, 2005]. One of the most prevalent academic studies at travel pricing is the trading agent competition (TAC). At this competition software agents compete for the best hotel rooms and airfare in a simulated environment [Wellman, 2004]. Although the study focuses on hotel room auctions, the most successful agents are those that also analyzed flight prices along with the hotel auctions. For example, some groups used price-equilibrium for supply and demand or neural networks to predict which flights were likely to affect hotel prices. For the flights each direction was sold separately. The initial price for each flight started randomly between $250 and $400 and followed a random walk with an increasingly upward bias. Thus choosing from the lower flight
prices earlier in the competition influenced which hotel auctions would result in the best combined price [Wellman, 2004].

2.1.5 Summary

This sub-section contains a summary of section 2.1, an airline industry review. The key to the difficulty in airline ticket price prediction is yield management. Yield management, or revenue management, attempts to predict consumer behavior to return optimal revenue. In the airline industry, the theory differentiates business and leisure customers by, for example, charging significantly more right before departure and charging less to customers whom stay over a Saturday night. The data for these systems are stored in central computer reservation systems, also known as global distribution systems. These databases store everything about a flight segment including route, date and times, carrier, jet, rules, and, of course, prices. It is extremely difficult to search this database for the cheapest airfare since there are a mind-boggling number of possible routes given a starting and destination pair.

Ticket purchasing tips follow. Tickets are cheapest at odd hours and weekends, particularly staying over Saturday night. Travel less popular routes and airports when possible. Multiple-leg flights and longer flights tend to be cheaper. Avoid holidays, Mondays, and Fridays when possible. The cheapest days to fly are Tuesday and Wednesday. Reserve a ticket at least 21-days in advance; though there is another category of tickets: last-minute fares. Airlines would rather have someone in a seat and have some additional revenue instead of an empty seat and lose that revenue since the
plane will fly either way. If applicable, check for student and senior discounts. In addition to searching for tickets online and through airlines, check with a travel agent. They can perform different types of searches with access to the source database. When watching prices, try to buy in a valley and try to avoid spikes. When ready to purchase, check back in an hour or two to make sure not in a spike. Knowledge of the volatility of the flight aids in buying on down turns. Reviewing patterns from similar previous flights, specifically the past year, can help understand ticket price patterns; for example, this information can be found at FareCompare.com. There’s a rumor that Wednesday at midnight is a good time to buy tickets. Tickets are loaded each weeknight at the same time, so Wednesday is not particularly cheap. Finally, when purchasing online, carefully watch the price charged as it could increase during checkout.

Following are Website Samples:

Airlines

www.AA.com
www.NWA.com
www.Continental.com
www.United.com
www.Delta.com
www.USAirways.com
www.Southwest.com – Southwest and JetBlue are low-cost carriers, search separately
www.JetBlue.com
Online travel Agents

www.Orbitz.com
www.Travelocity.com
www.Expedia.com

Meta-sites – Searches Airlines and Online Travel Agents

www.Kayak.com
www.Mobissimo.com
www.Sidestep.com
www.Farechase.com
www.Priceline.com – provides an auction
www.Hotwire.com – provides an auction

Airline Information

www.Farecast.com – provides predictions of price changes from current offerings
www.Farecompare.com – past year of flight history: high, low, and average ticket price
www.Tripstarter.com – flight history for current and previous year, highlighting cheapest months and holidays
www.Flyspy.com – gives chart of offerings over the next 30 days, including nearby airports

Deals

www.Site59.com – last minute deals
www.Cheapflights.com – lists of airline deals
www.Airfarewatchdog.com – lists of airline deals
In relation to the hypotheses, the tips given by industry experts lead one to question the source of the tip. Yield management makes this problem difficult. The tracking of spikes and weekly price swings attracted study and encouraged algorithm modifications.

This section overviewed the airline industry, ticket purchasing suggestions, online shopping, and a new feature in the online airfare shopping industry: airfare prediction. The techniques that this prediction technology uses will be discussed in the next section, preceded by an overview of airline travel analysis from a customer perspective.

2.2 Airline Analytical Perspective

The previous section gave an overview of the airline industry. This section contains an overview of an analytical perspective as well as a description of Etzioni’s work with Hamlet.

2.2.1 Overview

This section looks at the airline industry from an analytical perspective. The theoretical problem of finding airline tickets through all of the possible routes and possible airlines is a hard problem. Mathematician Keith Devlin [2006] explains:

“the (idealized) problem of finding the cheapest airfare between two given
locations is actually unsolvable, and even if you specify the actual route or the flights, the (idealized) problem of finding the lowest fare is NP hard, which means it could take the fastest computers billions of years to solve. The problem was that all of the different pricing rules interact in ways that not even those who designed the pricing systems could begin to fully understand. Mathematically, this made the (idealized) problem of finding an optimal fare between two given locations undecidable, which means that it is impossible to write a computer program to solve the problem. For the same reason, the more specific (idealized) problem of finding the optimal fare for a particular route, while theoretically solvable, turns out to be very similar to a classical mathematical problem known as boolean satisfiability, which has long been known to be NP-complete – which means it could take the fastest computer longer than the lifetime of the universe to find the solution. You can get a sense of just how complicated the real situation is when you consider that with airplanes offering thousands of different fares, with different sets of rules governing the different legs on each trip, if two people take a round trip together, with three flights in each direction, there can be as many as $1000^{12}$, or around $10^{36}$, fare combinations.” [Devlin, 2006].

Carl de Marcken [2003], Chief Scientist and co-founder of ITA Software, describes some of the airline industry’s mathematical characteristics. He illustrates that the industry uses a hub-and-spoke system, where the hub is a major airport,

“averaged uniformly, each airport has an outgoing degree of 8 (it has flights to 8 other airports), and is served by 4 airlines. However large airports dominate the
system: re-weighted by their number of departures, airports average degree 64. For a given airline, one to four airports account for half of their departures” [de Marcken, 2003].

Carl de Marcken [2003] continues with an example of the complexity of ticket searching:

“At 30,000,000 flights per year, standard algorithms like Dijkstra’s are perfectly capable of finding the shortest paths. However, as with any well-connected graph, the number of possible paths grows exponentially with the duration or length one considers. Just for San Francisco to Boston, arriving the same day, there are close to 30,000 flight combinations, more flying from east to west (because of the longer day) or if one considers neighboring airports. Most of these paths are of length 2 or 3. For a traveler willing to arrive the next day the number of possibilities more than squares, to more than 1 billion one-way paths. And that’s for airports that are relatively close. Considering international airport pairs where the shortest route may be 5 or 6 flights there maybe be more than $10^5$ options. One important consequence of these numbers is that there is no way to enumerate all the plausible one-way flight combinations for many queries, and the (approximately squared) number of round-trip flight combinations makes it impossible to explicitly consider, or present, all options that a traveler might be interested in for almost all queries” [de Marcken, 2003].

Next, de Marcken [2003] explains why the ticket pricing is so complex with so many
fares and such complicated rules. If the airline charges a small fee per ticket, the plane will fill, but the revenue will not cover the fixed costs of the flight. Similarly, if the airline charges a large fee, only a small number of people are willing to fly at that price, so the revenue will again not cover the fixed costs of the flight. The tough balancing act is to get those who are willing to pay higher prices to pay them. The airlines solve this problem in two ways, collectively called revenue management, or yield management.

Carl de Marcken [2003] illustrates the two solutions:

“The first is to use fare prices and fare rules to construct a system where the cheapest fares have restrictions that increase their perceived cost for a business traveler to the point where the business traveler will chose to buy more expensive fares. For example, cheap fares require round trip travel, prohibit non stop and ticket refunds, et cetera. But the cheap fares remain available for leisure travelers with more flexibility, for whom the extra restrictions are not so onerous. The second way … consists of dynamically deciding whether to sell a given fare for a flight based on how much demand there is for a flight. For example, if a flight is not filling, lower priced fares are made available (on the grounds that it’s better to get some money than none) but on high-demand flights only the most expensive fares are available. Importantly, the information the airline uses to estimate demand changes constantly, so seat availability responses may fluctuate up and down even in absence of any reservations” [de Marcken, 2003].

See figure 8 for an example of seat availability fluctuations as departure approaches. The y-axis shows different fare classes for a flight. The chart shows sample behavior of availability for a fare class as departure approaches. Figure 8 shows that the availability
of tickets in the F and Y classes decrease steadily as departure approaches. Class H begins to steadily decrease, returns to full price, jumps back down, then steps down to minimum price, and finally steps back up twice for the final days before departure. The cheapest coach class in this example takes a step down and then drops to minimum availability at just over a month out with a short return to near the previous availability. The final ticket class G where the revenue from having a customer sit in the seat is higher than the incremental cost of that customer. Note: this figure does not show when ticket availability is completely cut off.

![Availability Dynamics](image)

Figure 8: Availability Dynamics example for different fare classes [de Marcken, 2003]

This section covered some of the theory and behavior of airfare pricing. The next section discusses methods used in practice for price prediction.
2.2.2 Price Prediction Methods Employed In Practice

One area where price prediction has been more extensive is auctions. Sellers can benefit from algorithms that predict the final selling price of an item based on similar past auctions. Ghani [2005] put together three types of algorithms to approach this problem: regression, decision trees, and neural networks. Different forms of regression were used: linear regression, polynomial regression with degrees 2 and 3, and CART (Classification & Regression Trees). Decision trees and neural networks were used in two different ways: as multi-class classification and multiple binary classification tasks. A multi-class technique looks at classifying the data into groups, for example, in $5 increments. Alternatively, multiple binary classification looks at whether or not an item belongs to a particular class. In the paper, Ghani [2005], they found it was better to use a range rather than disjoint classes in the multiple-binary classification tasks; for example >$40 and >$50 are two classes and it is separately determined whether particular data items belong to both, either, or neither class independently. CART is a variation of a decision tree algorithm [Lewis, 2000]. These topics will be reviewed in detail later. Similar regression work can be found in “The Right Auction at the Right Price” [Reindorp, 2006] and “Predicting the final prices of online auction items” [Xuefeng, 2005].

2.2.3 Hamlet

The new airline ticket cost prediction website, Farecast, is particularly interesting to this thesis because its creator, Professor Oren Etzioni [2003], was the author of the major paper inspiring this thesis: “To Buy or Not to Buy: Mining Airline Fare Data to Minimize Ticket Purchase Price” [Etzioni, 2003]. Etzioni [2003] created Hamlet, which uses a
combination of data mining techniques to predict airline ticket prices: rule learning, Q-learning, and moving average as inputs to a stacked generalization.

2.2.3.1 Rule Learning

This section describes advancements in rule learning techniques and a gives a sample algorithm.

2.2.3.1.1 Rule Learning Overview

Etzioni [2003] uses Cohen’s [1995] rule learning system called RIPPER. RIPPER is adapted from decision tree learning. Decision tree learning classifies the data by identifying class differences along one dimension and iteratively splitting the data along different dimensions. Most algorithms generate, or grow, rules until each branch of the tree is completely classified, which leads to overfitting the training data. Once the growing is complete, extraneous rules are identified and pruned. Different approaches have looked at algorithms for intelligently growing the tree and improving the pruning. RIPPER improves on earlier work by using a better measure for the value of rules during pruning, using a different heuristic to determine when to stop adding rules to the set, and add a post-processing step after the rule set is grown to improve performance on the pruning step by considering alternatives to the existing rules [Cohen, 1995]. RIPPER [Cohen, 1995] is an improvement of IREP [Fürnkranz, 1994] which is an improvement of REP [Brunk, 1991].
2.2.3.1.2 Reduced Error Pruning

There are two phases to the Reduced Error Pruning (REP) [Quinlin, 1990] technique: rule learning (growing) and rule pruning. This technique uses a concept of training and test sets, where the training set is approximately 75% of the available data. Here, growing is done with the FOIL technique [Quinlin, 1990]. There are two operations in this technique: \textit{add-clause} and \textit{add-literal}. Rule and clause are synonymous. \textit{Add-clause} creates new clauses until all positive examples are covered by some clause. \textit{Add-literal} adds literals to a rule until the cause does not cover any negative examples. The literals are chosen to add to a clause by choosing the one with the most information gain; that is, the maximum portion of positive and maximum portion of negative examples. Once a clause is complete, that is it does not cover any negative examples, the positive examples are deleted from the training set. The growing phase terminates when there are no more positive examples [Quinlin, 1990].

The second phase of REP is rule pruning. The first phase extremely overfits the data. There are two operations in this phase: \textit{drop-last-literal} and \textit{drop-clause}. The rationality for \textit{drop-last-literal} is the most recently added literal has the lowest information gain. The rule pruning proceeds as follows: for each clause, temporarily apply \textit{drop-last-literal} and check against the test data set whether or not the accuracy increases. If it does increase the accuracy, make the drop permanent and move to the next literal. In addition to this, \textit{drop-clause} is applied similarly: for each clause, temporarily apply \textit{drop-clause} and test whether or not an increase in accuracy occurred against the test set [Quinlin, 1990].
2.2.3.1.3 Incremental Reduced Error Pruning

Incremental Reduced Error Pruning (IREP) [Fürnkranz, 1994] improves on some of the short-comings of REP. The algorithms are similar given the following changes:

1. The growing step and the pruning step are merged: after each clause is generated, it is tested for pruning. Each literal is testing for pruning by checking the accuracy of the proposed prune against the test set.

2. After a clause is added, all covered positive and negative examples are removed from the training set. The training and test sets are then again randomly selected.

3. The algorithm is terminated when a generated clause is pruned because the accuracy of the rule set against the test set is lower than without the rule. This rule has an inherent problem in that it will often lead to underfitting. Adding a less than ideal clause could have opened up other clauses, but this was not explored [Fürnkranz, 1994].

2.2.3.1.4 RIPPER

RIPPER (Repeated Incremental Pruning to Produce Error Reduction) [Cohen, 1995] is the rule learning algorithm used by Etzioni [2003]. RIPPER improves on IREP in the following ways.

1. The pruning metric during the growing phase changes. When a clause is generated, it is considered for pruning as follows: delete literals such that the number of positive examples plus the number of not covered negative examples is maximized. That is delete a literal that maximizes the following function until there are no more improvements
\[
\frac{p + (N - n)}{P + N}
\]

[Cohen, 1995]

where \( p \) is the number of positive examples for the clause, \( n \) is the number of negative examples for the clause, \( P \) is the total number of positive examples, and \( N \) is the total number of negative examples. A second pruning metric is used, maximizing the purity of a clause:

\[
\frac{p - n}{p + n}
\]

[Cohen, 1995]

2. The second change to IREP is to enhance the stopping criteria. IREP often stops too early. To improve the functionality, IREP encodes rules and terminates the growing when the difference between the lengths of the smallest and largest rules passes a (experimentally determined) threshold [Cohen, 1995].

3. Cohen [1995] proposes another improvement for IREP. At the completion of the original algorithm, an optimization step is added. For each rule in sequence, a replacement rule is considered by greedily adding literals to the rule, pruning like before, while testing for improved accuracy against the entire rule set. They call the final algorithm \( RIPPERk \) where the optimization step is repeated \( k \) times [Cohen, 1995].

**2.2.3.1.5 Rule Learning Algorithm**

A sample rule learning algorithm is given:
1. Given a data set, split it into a training set and a test set, where the training set is 3 to 4 times the size of the test set.

2. Create a new clause. For each literal, calculate the maximum portion of positive and maximum portion of negative examples. The literal with the maximum value is selected for growing.

3. Repeat step 2 until the clause does not cover any negative examples.

4. Test the literals for pruning: calculate the result from equation 1 and the purity of the rule by equation 2. For each literal in turn, consider its removal and delete the literal if the value of equation 1 gets worse or the purity decreases.

5. Add the pruned clause to the rule set.

6. Delete every positive and negative example covered by the added clause.

7. Randomly redistribute the training and test sets.

8. Repeat from step 2 until the difference in length between the longest clause and the shortest clause exceeds 64 bits [Quinlin, 1990; Fürnkranz, 1994; Cohen, 1995].

2.2.3.2 Q-Learning

Etzioni [2003] employs Q-learning, a type of reinforcement learning discussed by Sutton and Barto [1998]. This is a machine learning technique to push an agent towards a desired direction and away from an undesired direction. Imagine an agent is state $s_i$ with available actions $a_1, a_2, \ldots, a_k$. Upon performing one of the available actions, the agent is now in state $s_j$ with pre-defined reward or penalty associated with arriving at that state.
For each state $s$ and for each action $a$ at time $t$ the Q-learning formula is:

$$Q(s, a_t) \leftarrow R(s, a_t) + \gamma \max_{a'} Q(s_{t+1}, a_{t+1})$$  \hspace{1cm} (3)$$

[Etzioni, 2003]

where $R$ is the reward at a given state arrived by a given action, where $\gamma$ is the learning rate such that $0 < \gamma < 1$, and $\alpha$ is the discount rate for future rewards such that $0 < \alpha < 1$ [Etzioni, 2003; Q-Learning, 2007; Sutton 1998; Watkins, 1989]. $Q$, represented by a matrix, is initially empty and records the desirability to move to a state, action pair.

An algorithm using Q-learning could be as follows:

1. Initialize the Q matrix as empty.
2. Define a graph or matrix of the possible states and possible actions, including a goal state.
3. Define a matrix of the rewards for the defined states and actions.
4. Select $\gamma$ (in an advanced version of this algorithm this parameter may be adjusted through experimentation based on the quality of the results).
5. Select random initial state.
6. For each possible action from that state, select the maximum Q value.
7. Update the Q-value at that state.
8. Update the state to be the selected state.
9. Repeat from step 6 until the goal is reached.
10. Repeat from step 5 until a desired number of iterations is completed [Kardi, 2005].
2.2.3.3 Time Series Analysis

Additionally, Etzioni [2003] uses time series analysis discussed by Granger [1989]. Time series is a chronological sequence of data, typically spaced in uniform time intervals, used for modeling and prediction. There are three broad classes of time series analysis: autoregressive models, moving average models, and integrated models. An autoregressive model is a regression of the data, particularly to eliminate the noise in the data, giving the data’s trend [Autoregressive Moving Average Model, 2007]. A moving average is a set of means over ranges of \( n \) data points, often given an increasing weight as time progresses so that the most recent events have more weight, used to smooth out short term fluctuations [Moving Average, 2007]. An integrated model combines the autoregressive model and moving average model [Granger, 1989]. Specifically, Etzioni [2003] employs a moving average that looks at one week’s worth of data, weighting the more recent prices more heavily in the average using the following equation:

\[
p_{t+1} = \frac{\sum_{i=1}^{k} \alpha(i) p_{t-k+i}}{\sum_{i=1}^{k} \alpha(i)}
\]

[Etzioni, 2003]

where \( \alpha \) is a function with weighting chosen based on the problem, \( p_i \) is a price observation, and \( k \) is the number of points to average. Etzioni [2003] chose \( \alpha \) to be an increasing function of \( i \) and chose \( k \) to be a week’s worth of data. Etzioni [2003] used a simplistic prediction model: they look at the next time step predicted, and if the price decreases, purchase the ticket, otherwise wait. An inherent problem with moving average is the choice of \( k \); a higher value of \( k \) includes older data which gives a more
accurate average, but it also includes older data which could be pushing the estimate undesirably away from the newer data [Moving Average, 2007]. It is said that the moving average lags behind the newest data. Moving average could be used as follows:

1. Choose $a(i)$ appropriate for the problem.
2. Choose $k$, the number of to include in the average.
3. Choose the method in which to use the moving average as a basis of prediction. For example, the moving average increases, the moving average increased over the past $j$ steps, or some other problem appropriate method.
4. Wait until $k$ data points have passed.
5. Calculate the moving average for $k$ at the current time step.
6. Follow the curve created by the moving average and use the method determined in step 3 to decide whether to accept the prediction or return to step 5 at the next time step [Moving Average, 2007].

2.2.3.4 Stacked Generalization

Finally, Etzioni [2003] uses stacked generalization described by Ting [1999] to combine the three models described into one high-level model to achieve greater prediction accuracy. Each algorithm produces a different model, thus corresponding classes across models are examined and compared in an attempt to find the true members of the class [Ting, 1999]. Stacke1ed generalization is found to be successful in reducing the number of classification errors in the low-level models [Ting, 1999].
Basically stacked generalization has two levels of classifiers; the first set of classifiers results are validated by another classifier as follows. To improve the results of the first set of classifiers, J-fold cross validation is used. This is where the example data are broken into J (approximately) equal sets, where J-1 sets are used for training and the final set is used for testing. In the papers referenced, J is equal to 10. The classifier is run for each possible choice for the final test set and the resulting classes are averaged for the final result from one classifier in the first set. This is done for each classifier. The input to the validation classifier is the example and the output classification from each classifier used in the base set, and the output is the expected correct classification [Ting, 1999; Wolpert, 1992; Seewald, 2003]. It is up to the designer to decide on classifier choices based on appropriateness to the problem domain and data. Ting [1999] proposes using linear regression or decision trees for the level-1 classifier. Thus, the step by step algorithm could look like the following:

1. Choose the first set of classifiers, the base classifiers.
2. Choose the validation classifier, also called the meta-classifier or level-1 classifier.
3. Divide the data into J equal sets, where a good number for J is 10 by experimentation.
4. For one of the sets in the previous step, label it the testing set, and the rest of the sets together as the training set.
5. Run each base classifier against the sets as labeled in step 4 and get the resulting classifications.
6. Repeat from step 4 going through the list of sets, labeling them as the test set.
7. At this point, each example has a probability of the expected classification specified by each classifier. Use this as input to the level-1 classifier to determine the final classification for each example [Ting, 1999; Wolpert, 1992; Seewald, 2003].

The forerunner in airline ticket prediction, *Farecast’s Hamlet*, was discussed in this section. It uses a combination of four techniques for price prediction. The next section will discuss the technique used in this research for ticket purchasing: the Secretary Problem.

### 2.3 The Secretary Problem

The previous section gave an overview an analytical perspective of the airline industry, and also gave an overview of the earlier work this study is based on. This section contains the theory behind the Secretary Problem and a discussion.

#### 2.3.1 Secretary Problem Theory

The Secretary Problem, also known as the Sultan’s Dowry Problem or the Marriage Problem, is an optimal stopping problem where one is attempting to select the maximum candidate where candidates are considered one at a time in turn, where if a candidate is rejected, one cannot return to the candidate (i.e., they get another job). The decision must be made based on historical data (the candidates already rejected) and the current candidate. Ferguson [1989] states the Secretary Problem as:

1. There is one secretarial position available.
2. The number $n$ of applicants is known.
3. The applicants are interviewed sequentially in random order, each being equally likely.

4. It is assumed that the applicants can be ranked from best to worst without ties. The decision to accept or reject an applicant must be based only on the relative ranks of those applicants interviewed so far.

5. An applicant, once rejected, cannot later be recalled.

6. Nothing but the best: the payoff is 1 if the best candidate of the $n$ applicants is chosen and 0 otherwise [Ferguson, 1989].

The lack of ability to return to former candidates alludes to a strategy of picking the strongest candidate after some of them are rejected. The objective is then to find the ideal number of candidates to reject to give the maximum likelihood of success. A candidate randomly selected out of $n$ candidates has a probability of $1/n$ of being the best candidate. Plus Magazine [Smith, 1997] derives a result to improve those odds. $M$ candidates are rejected and some $Kth$ candidate is being considered, where $M-1 < K \leq N$ and $0 < M < N$ (see figure 9).

![Figure 9: K is somewhere between M and N](image)

The candidate is only considered if the highest so far was in the first $M-1$ of the $K-1$ already rejected. This happens with probability $\frac{M-1}{K-1}$, or in other words, the highest ranked candidate before the one being considered at $K$ is before $M$ with the given probability. Thus the overall chance of $K$ being the best out of $N$
is \( \frac{M - 1}{N(K - 1)} \). But \( K \) can take any of the values in the range from \( M \) to \( N \), so we can write: 

\[ P(M, N) = \sum_{K = M}^{N} \frac{M - 1}{N(K - 1)} = \frac{M - 1}{N} \sum_{K = M}^{N} \frac{1}{K - 1} \]

the choice of \( K \) cannot be two candidates at once, the probabilities are disjoint, therefore they can be summed [Smith, 1997]. The best value of \( M \) will be the one which satisfies: 

\[ P(M - 1, N) < P(M, N) \text{ and } P(M, N) > P(M + 1, N) \]

To find \( M \), we will solve both sides of the inequality.

Taking the first inequality, \( P(M - 1, N) < P(M, N) \)

\[ \frac{M - 2}{N} \sum_{K = M - 1}^{N} \frac{1}{K - 1} < \frac{M - 1}{N} \sum_{K = M}^{N} \frac{1}{K - 1} \]  

[Smith, 1997]

Removing the redundant factor of \( 1/N \) and rewriting the LHS:

\[ (M - 2) \left( \frac{1}{M - 2} + \sum_{K = M}^{M - 1} \frac{1}{K - 1} \right) < M - 1 \sum_{K = M}^{N} \frac{1}{K - 1} \]  

[Smith, 1997]

Similarly the second inequality, \( P(M, N) > P(M + 1, N) \) gives:

\[ M - 1 \sum_{K = M}^{N} \frac{1}{K - 1} > M \sum_{K = M + 1}^{N} \frac{1}{K - 1} \]  

[Smith, 1997]
We can use these inequalities to find $M$ and any $N$.

\[
1 > \frac{N}{K = M + 1} \sum_{K = M + 1}^{K} \frac{1}{K - 1}
\]  

[Smith, 1997]

\[
\sum_{K = M + 1}^{K} \frac{1}{K - 1} < 1 < \sum_{K = M}^{K} \frac{1}{K - 1}
\]  

[Smith, 1997]

\[
\frac{1}{M + 1} + \frac{1}{M + 1} + ... + \frac{1}{N - 1} < 1 < \frac{1}{M - 1} + \frac{1}{M} + ... + \frac{1}{N - 1}
\]  

[Smith, 1997]

Figure 10 shows a few sample values for $M$ and $N$.

<table>
<thead>
<tr>
<th>N</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

Figure 10: Sample values for M and N [Smith, 1997]

There is a sum in the calculation of $P(M, N)$ that appears in other situations in mathematics too. This is the $L$th harmonic number.

\[
H_n = 1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{n} = \sum_{k = 1}^{n} \frac{1}{k}
\]  

[Harmonic Number - Wikipedia, 2007]

\[
\sum_{K = 1}^{L} \frac{1}{K} \equiv \ln(L) + C \quad \text{Where C is Euler’s constant, approximately 0.577}
\]  

[Smith, 1997]

Using this equation, $P(M, N)$ is approximated as follows:
\[ P(M, N) = \frac{M-1}{N} \sum_{K=M}^{N-1} \frac{1}{K-1} = \frac{M-1}{N} \sum_{K=M-1}^{N-1} \frac{1}{K} \]  
(15)

\[ = \frac{M-1}{N} \left\{ \sum_{K=1}^{N-1} \frac{1}{K} - \sum_{K=1}^{M-2} \frac{1}{K} \right\} \]  
(16)

\[ \cong \frac{M-1}{N} \{ \ln(N-1) - \ln(M-2) \} \]  
(17)

\[ \cong \frac{M-1}{N} \ln\left( \frac{N-1}{M-2} \right) \]  
(18)

[Smith, 1997]

For large \( N \), the expression simplifies to:

\[ P(M, N) \equiv \frac{M}{N} \ln\left( \frac{N}{M} \right) \]  
(19)

[Smith, 1997]

In order to find the best value of \( M \), we approximate the conditions derived earlier:

\[ 1 < \sum_{K=M}^{N} \frac{1}{K-1} = \sum_{K=M}^{N-1} \frac{1}{K} = \sum_{K=1}^{N-1} \frac{1}{K} - \sum_{K=1}^{M-2} \frac{1}{K} \cong \ln\left( \frac{N-1}{M-2} \right) \]  
(20)

\[ 1 > \sum_{K=M}^{N} \frac{1}{K-1} = \sum_{K=M}^{N-1} \frac{1}{K} = \sum_{K=1}^{N-1} \frac{1}{K} - \sum_{K=1}^{M-1} \frac{1}{K} \cong \ln\left( \frac{N-1}{M-1} \right) \]  
(21)

[Smith, 1997]

Therefore, for large \( N \), the best \( M \) satisfies:

\[ 1 \equiv \ln\left( \frac{N}{M} \right) \]  
(22)
\[ \frac{M}{N} \approx \frac{1}{e} \]  \hspace{1cm} (23)

[Smith, 1997]

Also, \( P(M,N) \approx 1/e \). Thus, we have derived two conclusions: the best value of \( M \) given \( N \) is approximately \( N/e \) with a chance of success also \( 1/e \) or about 0.368 [Smith, 1997].

2.3.2 Secretary Problem Discussion

In summary, for each site and route combination, there are, say, \( N \) data points. The initial period of examining and rejecting the candidate pool will end after \( N/e \) candidates have been examined. That is, after 36.8\% of the candidates have been examined and rejected, a candidate will be selected that is better than the best candidate within that initial period. There is a 36.8\% chance this is the best candidate. Further, if no candidate is selected before the final candidate is reached, the final candidate is selected.

The result from the Secretary Problem, also known as optimal stopping, is used in problems where one’s objective is to select the best candidate where the value of each candidate is not known before the candidate is viewed. Two additional applications of optimal stopping are in selling real estate [Ferguson, 2000] and in auctions [Guo, 2002]. A house is being sold and offers come in daily. As the owner receives offers, the owner contemplates whether to accept the current offer, or reject it and wait for a better offer. When the owner decides to wait, example costs are house maintenance and the passage of time [Ferguson, 2000]. Similarly, in auctions, the seller places an item up for bid and
receives bids over time. In certain auction formats, the seller may accept a current bid offer and close the auction or wait for additional bids to arrive [Guo, 2002].

2.4 Comparative Results

Follows is an overview of interesting papers related to this work on the Secretary Problem. The focus of this review is to find applications of the Secretary Problem and Optimal Stopping theory found in this thesis. These applications are for background material and comparison to this thesis. This section may be skipped if the reader would like to skip ahead to the results.

2.4.1 Subject Tendency in Secretary Problems

There are many studies of comparing the theoretical solution to the Secretary Problem against the behavior of subjects in the same problem. Many papers find that subjects stop searching too early, but they often get within 10% of the optimal value in many cases and within 25% in other cases. The papers discuss the results of when and why subjects tend to terminate early, when it is a good decision and when it is not, and why subjects are often close to the optimal solution. Examples: Bearden [2007], Bearden [2006a], Bearden [2006b], Bearden [2005], Cox [1989], Dudey [2001], Lee [2007], Seale [1997], Todd [2007], and Zwick [2003].

2.4.2 Secretary Problem theory and no practical application

This class of papers extend Secretary Problem theory by relaxing or changing a constraint, but contain no applications or examples of their work, only theory. Theory is
wonderful, but this thesis needs an application for comparison. One such article extends previous work in generalizing the classical Secretary Problem where the payoff is dependant on the rank of the candidate selected rather than only selecting the best [Bearden, 2007]. That one looked for the maximum rank while Moriguti [1993] looked at the minimum rank. It has been proven that the cutoff for this generalization is the same as the classical case by Mucci [1973]. Ramsey [2005] looks at the case where two employers are looking at one candidate, and if the first employer does not select the candidate, the second employer has the option of hiring the candidate. Examples: Atjai [2001], Bearden [2007], Ferguson [1989], Freeman [1983], Gilbert [1966], Kao [1999], Lindley [1961], Lorenzen [1981], Lorenzen [1979], Moriguti [1993], Mucci [1973], Ramsey [2005], Shepp [1969], Szajowski [2002], Tamaki [1991], Tamaki [1986a], and Tamaki [1986b].

2.4.3 Continuous Secretary Problem

One pervasive extension of the Secretary Problem work here is the application to continuous domains. Tickets are examined discretely, so this body of work is not applicable. Some continuous problems may be modeled discretely; these are considered in a later section as applicable. Most of these examples are from finance. Examples: Pendersen [2007], Sunzu [2007], and Szajowski [2006].

2.4.4 Stock / Financial

The papers on the stock market lead to the conclusion that much of that work is not compatible with this investigation. Primarily, the stock market is continuous which
lends itself to other analysis. While the stock market could be modeled discretely, this research is not prevalent. Alternatively, there are other problems in the finance area that could be relevant to this topic, such as when a deadline exists and candidates are considered one at a time. Albert [1970] is an example of a deadline of April 15, tax day, where they must sell some of their investment to pay for capital gains; but, he uses his own dynamic programming approach. Boyce [1970] is a discussion of finding the optimal time to sell a bond that will mature in a few months guided by a model of the future predicted price. Cook [1992] discusses a deadline of January 1, 1987 when the tax rate was increased, and how market investors reacted to the news by timing the sale of an enormous quantity of stock. Examples: Albert [1970], Battauz [2004], Boyce [1970], Cook [1992], Chen [1999], Finster [1983], Griffeath [1974], and Jagannathan [2002].

2.4.5 Auction

Auctions are one area where this work is also applies, in special cases. Hajiaghayi [2004] describes that if agents bidding appear, bid, and depart disjoint from one another the Secretary Problem theory is directly applicable. He then extends the theory for a more general case. Examples: Hajiaghayi [2004], Kleinberg [2004], and Ulrich [2006].

2.4.6 Conclusion

The above papers highlighted examples from different areas in the literature that the Secretary Problem is found. The first area studies human psychological experiments to understand how subjects behave in problems such as the Secretary Problem. While the focus of their work is not relevant, the background they go into while introducing the
topic is exactly relevant. Their background and theory sections do contain interesting discussion. Some of them show examples of a random set of candidates, their relative ranks, and what the optimal selection would have been given their study of the problem [Seale, 1997; Bearden, 2006b; Bearden, 2007]. The second area here listed examples where mathematical extensions to the Secretary Problem theory are discussed, but no actual examples are given. These are not interesting to this study. The next two areas are more interesting, as they cover interesting applications of Optimal Stopping theory. While the most interesting topic in the stock area – finding the perfect, general trading price – is not applicable to this thesis topic because airline tickets have a deadline for purchase while stocks have an infinite horizon. There are other interesting topics in the finance area – when there is a deadline. Albert [1970] covers a problem where stocks must be sold by April 15 to cover capital gains. Boyce [1970] covers a problem if and when to sell a bond that is about to mature. Cook [1992] looks at the all of the stock sales right before the tax rate was increased in the on January 1, 1987. On a different note, auctions are another area where Optimal Stopping is applied. Hajiaghayi [2004] looks at predicting the highest bid for an auction by using the solution to the Secretary Problem. Among others, Hajiaghayi’s [2004] work is based on the Secretary Problem theory and has results comparable to this thesis.

2.5 Secretary Problem Variations

This section contains a review of variations to the Secretary Problem and discusses whether or not they are relevant to this study. The following is based on Freeman [1983], Rose [1984], Gilbert [1966], and Atjai [1995].
2.5.1 Not Relevant

This sub-section contains the variations to the Secretary Problem that are not relevant to this thesis.

2.5.1.1 Recall

Recall is returning to accept a previously rejected candidate [Yang, 1974]. Recall is not relevant to this study since once a ticket price is missed, the ticket price offered at that time is no longer available.

2.5.1.2 Random Arrivals

Random arrival of candidates for consideration has been studied by Karlin [1962]. In the case here, while the rank of the next candidate is not known until it is observed, candidates are observed at pre-determined times.

2.5.1.3 Multiple Selections

Gilbert [1966] was the first of many to look at accepting multiple candidates. While an interesting topic, the goal of this study is the purchase of a single ticket, so this research is not relevant.

2.5.1.4 Limited Memory

One variation of the Secretary Problem is to only have enough storage for one candidate. This is possible because only the apparent rank of the best candidate so far
needs to be stored [Rubin, 1977]. Though, the assumptions of this study do not cover limited storage.

2.5.1.5 Continuous

There is a subfield of study when the candidates are viewed in a continuously changing medium [Dynkin, 1969]. This subfield is not relevant because candidates are viewed discretely in this study.

2.5.1.6 Opponent

There are two variants of an opponent changing the sequence: 1) an opponent can order the candidates in any way desired, and 2) ahead of time, an opponent can replace one candidate as the best candidate in an attempt to foul the player’s approach to the problem [Gilbert, 1966]. While the airline industry would like to charge the highest price any given consumer is willing to pay, forces outside of the control of the airline influence the ticket prices; namely, the demand for a particular flight can only be predicted and it is not known ahead of time. The price offered at any time before departure is influenced by a system trying to categorize the shopper as a business or leisure traveler and adjust the price of the ticket accordingly by looking at, for example, whether or not there is a Saturday stay-over and if the ticket is being purchased either weeks or only days ahead of time. The current airline ticket pricing strategy is too volatile to be covered by research in the area of an opponent manipulating all or one of the tickets ahead of time.
2.5.1.7 Full Information Search

There is an area of study concerning knowing as much as possible of the following candidates. That is, the distribution is known as well as the properties of the distribution. For example, for the normal distribution, the mean and standard deviation are known. In this case, all that is not known is the actual following candidates. An extreme example given by Gilbert [1966] follows: suppose that there are 10 candidates and the first candidate has a measurement of 0.9998 out of 1. The chance that the other 9 numbers are smaller is \((0.9998)^9 = 0.99982\) [Gilbert, 1966]. Thus, the first candidate would be chosen without consideration of the remaining candidates. The study in this thesis is not nearly as extreme as this case, as there are numerous unknown, hidden influences on ticket prices. The following price is often in the neighborhood of the preceding price, but frequently enough prices change suddenly without warning. Sometimes they return to an earlier price, and other times they do not. Hidden factors such as seat availability at any given time make it difficult to determine the trend. Airline ticket price search is certainly not a full information problem.

2.5.2 Relevant

The following topics are relevant to this application of the Secretary Problem. How they could have been applied is discussed.

2.5.2.1 Some Probability of Not Being Available

When purchasing airline tickets, there is some probability of the ticket not being available when the decision to purchase a given ticket is made. This happens when
multiple consumers wish to purchase the same ticket and another consumer purchases the ticket first. Smith [1975] is the first author to cover the case when the candidate is not available. A similar derivation as covered earlier is followed, except an extra term, $p$, is added for the ticket availability probability as follows (recalling that $M$ is the candidate and $N$ is the number of possible candidates).

$$v = \frac{p}{(1-p)N} \left[ (M - p) \prod_{k=M}^{N-1} \left( 1 + \frac{1-p}{k} \right) - M + 1 \right]$$

(24) [Smith, 1975]

$$\lim_{N \to \infty} \frac{M}{N} = \lim_{N \to \infty} v = p^{1/(1-p)}$$

(25) [Smith, 1975]

This yields approximately a 25% chance of finding the best candidate when the probability of the candidate not being available is 50%, 34.867% when the probability of not being available is 90% and 36.787% when a candidate is always available, just as before [Smith, 1975]. In this study, the chance that a ticket is not available was not taken into account. If it had been, when the best candidate was expected to be found, there would have been an appropriate probability that it would not be available and the search would continue until a ticket was successfully purchased or the end of time was reached. One would expect that this would end up with worse prices in general, though this might allow for some better ticket pricing as well when the chosen ticket was not actually the best ticket and a better one is found later.
2.5.2.2 Two or More Competing Decision Makers

One angle to consider is the theory where two decision makers are competing for the same candidate. Unfortunately, based upon review and the surveys by Abdelaziz [06] and Sakaguchi [94] a paper is not found reviewing the situation here: the basic Secretary Problem, without any assumptions generalized differently than covered here, where two decision makers are viewing the same candidates, without recall, known N, unknown distribution (no information), and the decision makers are not working together where one has priority over the other. Thus, a proposed workaround to this problem is to leverage the earlier discussion on a candidate not being available and determining the probability empirically.

2.5.2.3 Minimizing the Expected Rank

In the basic Secretary Problem, only the selecting the best is rewarded. To be able to apply this theory to more general problems, this is relaxed and the utility of the $i$th best item takes the value of $n - i$ [Freeman, 1983]. The probability that the $s$th best item chosen out of the first $r$ items has true rank $i$ out of $n$ items is

$$p_{i,n}(r,s) = \binom{i-1}{s-1} \binom{n-i}{n} \binom{r-s}{r} \quad (i = s, s+1, ..., n+s-r) \quad (26)$$

[Lindley, 1961; Freeman 1983]

with expected utility

$$U(r,s) = \sum_{i=s}^{n+s-r} (n-i) p_{i,n}(r,s) = n - \frac{n+1}{r+1} s \quad (27)$$

[Lindley, 1961; Freeman 1983]
yielding dynamic programming equation

\[ V(r, s) = \max \left\{ U(r, s), \frac{1}{r+1} \sum_{s'=1}^{r+1} V(r+1, s') \right\} \]

(28)

\[ V(n, s) = n - s \]

(29)

[Lindley, 1961; Freeman 1983]

Interestingly, Chow [1964] first showed that as \( n \to \infty \)

\[ V_n = V(0,0) = V(1,1) \to \prod_{j=1}^{\infty} \left( \frac{j + 2}{j} \right)^{1/(j+1)} = 3.8695 \]

(30)

[Chow, 1961; Freeman 1983]

or in other words, the expected rank of the chosen candidate as \( n \to \infty \) is 3.8695 [Chow, 1964].

2.5.2.4 Observation Cost

Interacting with any system has a cost. Here, the majority of the cost is the time it takes to look up the prices. There are also minor costs such as the storage space, computer resource, internet connection, power, code maintenance, and monitoring. These costs are basically fixed and Lorenzen [1979] proved that this does not alter the search strategy. It is significantly more complicated when the observation cost varies. The case when the cost increases as the time advances has been analyzed in detail. That case is not applicable to this study because the costs here are fixed.
2.5.2.5 Discounted Gain

Rasmussen [1975] covers an area called discounted gain. The concept of discounted gain follows from the earlier observation cost discussion, but now the cost increases as each candidate is observed. This is to simulate the case where it is desired to choose a candidate earlier in the series of observations. For \( 0 < \alpha \leq 1 \), the reward is \( \alpha^k \) if the best candidate is selected, otherwise 0 [Rasmussen, 1975]; this is similar to the basic problem with the reward reduced the longer the selection continues. When \( \alpha = 1 \), the results are the same as before. The technique is the same as before: the first \( r - 1 \) candidates are reviewed and immediately rejected, recording the candidate with the highest relative rank so far. The candidate accepted is the next candidate with a ranking higher than all of the candidates seen before the boundary. The difference here is in the selection of \( r \) for the boundary; the \( r \) chosen is the smallest one that satisfies

\[
\{ r \geq 1 \mid \sum_{k=r}^{N-1} \alpha^{k+1-r} / k \leq 1 \}
\]

[31]

[Rasmussen, 1975]

where \( N \) is the number of candidates, \( \alpha \) is the discount, and \( k \) represents the possible candidates [Rasmussen, 1975].

This was not considered in this study because the focus was to gather and analyze the data without consideration for the cost to gather the data over time. Additionally, the goal here is to choose the lowest price possible without consideration to the time at which it is purchased. If such a discount factor were considered, the same results would be reached, for the routes with approximately 1500 candidates, with an \( \alpha \) approximately
0.998 or greater and the routes with approximately 2000 candidates would have an \( \alpha \) of
approximately 0.99999 or greater. If the alpha is lower, the boundary would shift to the
left and the results could change if the original best candidate before the line is passed.
For example, in CVG-BZN, if the \( \alpha \) falls from above approximately 0.99999 to below
approximately 0.99965, the best price before the boundary drops from $453.70 to $603.7
and thus the Secretary Problem result is $453.70 down from $1337.70, the last price in
the series. An additional difficulty is determining the appropriate \( \alpha \) given each
consumers’ situation and values.

2.5.2.5 Backward Induction

Ajtai [1995] completed work on an alternate approach to determine the best
candidate. Though his work is intended for selecting a general number \( k \) candidates, the
algorithm can also be applied to just one candidate. The setup and the background
follow. First, the sequential series of candidates is broken into
\[ m = \log n + 1 \]
consecutive intervals where the \( i \)th (\( i = 1, \ldots, m \)) interval \( I_i \) is

\[
I_i = 1 + n \sum_{j=1}^{i-1} 2^{-j}, \quad n \sum_{j=i}^{i} 2^{-j} \quad \text{if } 1 \leq i \leq m-1
\]

\[
\text{And } I_i = n \quad \text{if } i = m
\]

[Ajtai, 1995]

which is the same as saying that the first interval contains half of the elements and each
following interval contains half of the remaining elements except for the final interval
containing the last element [Ajtai, 1995]. The first \( m' \) intervals are called opening
intervals while the rest are closing intervals, where
\[ m' = \max \left\{ 0, \left\lfloor \log(k / p) \right\rfloor \right\} \]

where \( p = 64 + \log^2 k \)

[Ajtai, 1995]

Further, they say the minimum number of acceptances for each \( I_i \) is

\[
p_i = \begin{cases} 
  k2^{-i} & \text{if } i \leq m' \\
  k2^{-m'} & \text{if } i = m'+1 \\
  0 & \text{if } m'+1 < i \leq m
\end{cases}
\]

[Ajtai, 1995]

The algorithm tries to select the top \( k \) objects during a given interval, and also to make up for the number of objects that were expected but not selected in prior intervals [Atjai, 1995]. Thus, at each interval a threshold is set to select at least as many objects as required before this interval [Atjai, 1995]. Let \( Q_j \) (\( j = 0, \ldots, i-1 \)) be the number of items accepted in \( I_j \). Then \( D_{i-1} \) is the difference between number required in the previous interval and the actual number selected

\[
D_{i-1} = \max \left\{ 0, \sum_{j=1}^{i-1} p_j - \sum_{j=1}^{i-1} Q_j \right\}
\]

[Ajtai, 1995]

Then, the acceptance threshold for \( I_i \) is

\[
A_i = \begin{cases} 
  D_{i-1} + p_i & \text{if } \sum_{j=1}^{i-1} Q_j < k \\
  0 & \text{otherwise}
\end{cases}
\]

[Ajtai, 1995]

And each interval is expected to contain \( k/2^i \) of the top \( k \) objects [Atjai, 1995].

67
In the opening intervals, the algorithm tries to accept \( A_i \) with probability \( k^{-5(z+1)} \) objects, and an expected number of \( A_i6(z+1)\sqrt{A_i \log k} \) [Atjai, 1995]. In the closing intervals the algorithm attempts to accept \( 32(z+1)A_i \) objects with a probability \( 2^{-5(z+1)(a_i+1)} \) where \( a_i \) is some value of \( A_i \) [Atjai, 1995]. Finally, the algorithm does not accept any of the first \( \lceil n/(8\sqrt{k}) \rceil \) objects [Atjai, 1995]. The lower-bound on the sum of the \( z \)th powers of the ranks is \( kn^2z^{2(z+1)} \) and the expected value is \( k^{z+1}/(z + 1) + O(k^z) \).

The algorithm processes the items sequentially one at a time, and at the beginning of each interval calculates \( A_i \) as earlier described [Atjai, 1995]. For each item \( d \) in each interval:

1. If all \( k \) objects have already been chosen, the object is rejected.
2. Otherwise,
   (a) “If \( i \leq m' \), the \( d \)th item is accepted if it is one of the top \( \lceil (A_i + 6(z+1)(\sqrt{A_i \log k})2^d/n) \rceil \) items among the first \( d \)” [Atjai, 1995].
   (b) “If \( i > m' \), the algorithm accepts the \( d \)th item if it is one of the top \( \lceil 32(z+1)(A_i)2^d/n) \rceil \) items among the first \( d \)” [Atjai, 1995].
3. Otherwise, if the number of candidates remaining is the same as the number of items left, accept \( d \) [Atjai, 1995].

When there is \( k = 1 \) objects to be selected, and \( z = 1 \) for the simplest case, here is what the setup and algorithm looks like:

\[ p = 64 \]
\[ m' = 0, \text{ meaning there is one interval} \]
\[ p_i = 1, \text{ meaning there is 1 acceptance in the one interval} \]
\[ D_i = 1, \text{ meaning there is 1 left over not already selected} \]
\[ A_i = 1, \text{ meaning there needs to be at least 1 item selected in the one interval} \]

Algorithm: for each candidate \( d \),

1. If one candidate has been selected, then reject current candidate.
2. If \( \lceil n/8 \rceil \) candidates have been rejected, accept \( d \) if it is one of the top \( \lceil 128 \times d/n \rceil \) candidates.
3. If \( d \) is the final candidate, accept it.

For example, if \( n \) is 2000, the threshold is 250. For candidate 251, accept the candidate if its relative rank is among the top 62 candidates seen so far. The selected candidate’s absolute rank is expected to be at least 125. Based on the data from the study, for example Continental CVG-BZN, with 2093 candidates, the boundary would be at candidate 262 instead of 770, candidate 262 would be compared to the top 16 candidates, and it would have selected the 262 candidate at $458.70 when the secretary algorithm used in this study chose $389.80 in this case.

### 2.6 Data Analysis Techniques

The preceding three sections covered the nuts and bolts of the Secretary Problem. This section describes statistical and machine learning techniques commonly used for prediction and classification that could have been used but were chosen not to be employed for this study. This section may be skipped for those interested only in hypotheses coverage. A summary of technique comparisons is found in the next section.
2.6.1 Regression

This section covers linear regression and logistic regression.

2.6.1.1 Linear Regression

Linear regression is a model capturing the system in a linear function of parameters.

\[ Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \varepsilon \]  

[Linear Regression, 2007]

where \( Y \) is the dependent variable, \( X_p \) are the independent variables where \( p \) is the number of parameters to be estimated, \( \beta_k \) are the terms adjusted to fit the data, and \( \varepsilon \) is a random error term. The error term is used to represent unpredicted or unexplained variation.

Typically, the \( \beta \) terms are adjusted by training data using the method of least-squares. This is where the \( \beta \) terms are adjusted such that the sum of the residuals (errors) is minimized by sum of the square of the difference between the actual value in the training data and the value given by the equation up to this point. The equation is as follows

\[ S = \sum_{i=1}^{n} (y_i - Y)^2 \]

[Least Squares, 2007]

where \( y_i \) is the expected value for the training set at example \( i \). Since \( \beta_k \) are real values, finding the best \( \beta_k \) by trial and error to minimize the sum of residuals is not practical.

For the equation \( y = a + bx \), with \( n \) data points, \( a \) and \( b \) can be derived as follows:

\[ a = \frac{\left( \sum y_i \right) \left( \sum x_i^2 \right) - \left( \sum x_i \right) \left( \sum x_i y_i \right)}{n \left( \sum x_i \right) - \left( \sum x_i^2 \right) \right)} \]

(39)
\[
    b = \frac{\left( \sum x_i y_i \right) - \left( \sum x_i \right) \left( \sum y_i \right)}{n \left( \sum x_i^2 \right) - \left( \sum x_i \right)^2} \tag{40}
\]

[Principe, 1999]

2.6.1.2 Linear Regression Algorithm

This is a sample algorithm that might be used to implement linear regression:

1. Based on \( p \), algebraically derive the affect of each observation parameter on each \( \beta_0, \ldots, \beta_p \).

2. Using the first \( p \) points in the training set, solve the \( p \) system of linear equations to determine \( \beta_0, \ldots, \beta_p \).

   For each point remaining in the training set:

3. Adjust \( \beta_0, \ldots, \beta_p \) such that the sum of the residuals is minimized using the equations found in step 1.

4. Repeat from step 3 until the training set is exhausted [Linear Regression, 2007; Least Squares, 2007; Principe, 1999].

2.6.1.3 Logistic Regression

When it is not known whether the data follows a linear model, or when the independent variables discrete or categorical, it is preferred to use a logistic regression model [Connor, 2002; Agresti, 1996]. The model is:

\[
P = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p)}} \tag{41}
\]
where the parameters are the same as in linear regression. Here, the \( \beta \) parameters are trained using theory from maximum likelihood estimation. \( L(w|y) \) represents the likelihood, or the probability, of outcome \( w \) given the observed data \( y = y_1, y_2, \ldots, y_n \). The calculation of \( L \) depends on the probability distribution of the data. The simplest case is where the outcome is binary which has the binomial probability distribution:

\[
L(w \mid n, y) = f(y \mid n, w) = \frac{n!}{y!(n - y)!} w^y (1 - w)^{n-y}
\]  

(42)

[Myung, 2003]

where \( y \) is the observed data, \( n \) is the number of observations, and \( w \) is the desired outcome. Since the goal is to maximize \( w \), the probability of the desired outcome, it is left as a variable and the maximum of this function is found. For example, in a sequence of 10 Bernoulli trials the number of successes is set to 7. In this case:

\[
L(w \mid n = 10, y = 7) = \frac{10!}{7!3!} (w)^7 (1 - w)^3
\]

(43)

[Myung, 2003]

generates the graph in figure 11. In this case, it is most likely that the success rate on each trial was 0.7 with a probability of 0.267. The catch is that in this equation \( y \) is a value representing an observation. For parameterized observations Myung [2003] states that it is usually not possible to obtain an analytic solution and numerical methods must be used.
2.6.2 Neural Networks

This section gives an overview of neural networks and a sample algorithm.

2.6.2.1 Neural Networks Overview

This machine learning system was designed to simulate the neural activity in the human brain. The brain is composed of millions of neurons which receive input from multiple sources, process the input, and sends the output to the next neuron. Similarly, a machine learning technique is made up of layers of neurons, where the first layer received the input from outside of the system, the last layer transmits the output from the system, and there are some number of intermediate layers in between the input and output layers.

![Figure 11: L(w|n=10,y=7) graph [Myung, 2003]](image-url)
These layers are called hidden because they do not directly affect the output and are not directly affected by the input. A sample neural network structure is shown here:

![Sample Neural Network](Artificial_Neural_Networks_2007.png)

The nodes in the system represent neurons. Each of these nodes maintains a set of weights; one weight for each input. These weights modify the inputs from the previous layer to be sent to the following layer if the sum of inputs times the weights pass a threshold. The signals sent from the previous layer were modified by the weights at that layer of nodes, and so on. This complex network of interactions leads this technique to be stated as a black box system; that is, it is not clear how the differences in the independent variables affect the parameters at each node to predict the dependent variable at the output [Yoo, 2005]. Neural networks are able to approximate any arbitrary function by some number of hidden layer neurons, but it is not understood how to determine how many nodes to use at the hidden layer [Vaccari, 1994].
The weights at each node are set by training the system. The type of problem at hand and the data available will dictate the training methodology. There are three classes of learning: supervised, unsupervised, and reinforcement learning. Supervised learning is when a set of training data is available with the correct classification given with the data. These techniques are used to build a system to predict correct classifications of future data seen where the classification is unknown beforehand. Unsupervised learning is where the correct classification is not known ahead of time; only a sample of data is available. These techniques break the data into natural classes based on differences in the dimensions of the sample data so that future data are recognized as belonging to one of those groups. Reinforcement learning is a blend between the two techniques, albeit closer to supervised learning. Reinforcement learning requires of the correct classification of a set of training data. When a technique believes a particular item belongs to one class, but it actually belongs to another class, a measure of the error between the correct and actual classifications is used to prompt the technique to lean more towards the correct classification.

The neural network training algorithm follows the pattern of reinforcement learning. It begins by assigning essentially a random assignment of weights for each input on each node. The training data is then passed through the network. When the network incorrectly classifies an example, the error from the correct classification is calculated and the error is propagated back through the network in such a way to reinforce the weights to be closer to the correct classification the next time it sees similar examples. This method is called back-propagation. The error \( \text{error} = (d - o)^2 \) where \( d \) is the desired
output and $o$ is the current output. Thus, the goal of back-propagation is to minimize the sum of errors over all of the training samples:

$$\text{Error} = \sum (d - o)^2$$

[Clabaugh, 2000]

This leads to running all of the training data through the neural network many times until the error is acceptably low. It is not desired for the error to be too low or the training set will be overfit and not general to the all possible data.

The error at the output is propagated to the middle layer, multiplied by the weight to get back to the node of interest. When going to the input layer and calculating the error, the error factor is calculated by summing the error factors times the weight from all of the nodes affected by the node currently being calculated.

The basic back propagation algorithm [Bernacki, 2005] to update the weights is $w_{ij}' = w_{ij} + \eta \delta_{k} x_{i}$ where $\eta$ is some learning rate and $w_{ij}$ is the weight at node $i$, input $j$ with error $\delta_{k}$ at node $k$. The learning rate could be high at the start to imply high learning and then reduce as the iterations pass.
2.6.2.2 Neural Networks Algorithm

Here is one possible algorithm for using a neural network.

1. Choose the number of hidden layer neurons.
2. Set the number of input neurons as the number of independent variables.
3. Randomly assign weights to all of the inputs to every node.
4. Determine the learning rate change function.
5. For each training vector: For each layer in the network: For each node in the layer: For each input dimension: multiply the input by the weight; if the result is higher than the threshold, propagate a 1 to the next layer, otherwise propagate a 0 to the next layer.
6. Calculate the error of the output of the network by taking the difference of the network output and the expected output squared.
7. The output error is propagated to the hidden layer nodes by multiplying by the node weight for each node.
8. The error at the input layer nodes is taken as the sum of the errors for each path multiplied by the weight along that path.
9. The weights are then updated by \( w'_{ij} = w_{ij} + \eta \delta_{ik} \).
10. Repeat from step 5 for each training vector.
11. Repeat from step 5 until satisfied with the resulting error rate; also update the learning rate.

2.6.3 Decision Trees

This section reviews decision trees.
2.6.3.1 Decision Trees Overview

The decision tree model is a classification technique. It is popular because it is simple to understand and interpret because of its visual representation, as seen in figure 12, an example of a decision tree a financial institution could use to determine whether or not a person should be offered a loan. To decide if someone should receive a loan, start at the root of the tree and follow the branches of the root that corresponds to the applicant.

![Decision Tree Example](image)

Figure 12: Decision Tree Example: decide whether a person should be offered a loan [Wilson, 2006]

The children of a node represent a differentiation of the classes represented at that node along a particular dimension, or variable. The leaves of the tree represent a particular class, and it has to the properties of the branches of the tree followed to get there. Another way of thinking about it is that the leaves represent a class with the dimensions set to particular values, the ones found along the path to get from the root to the leaf.
To systematically generate a decision tree, one needs a set of training data composed of the desired classes where a class is composed of representative data. One begins with the entire set of data at the root of the tree. The data are then split into the children nodes by one of the techniques described shortly. These techniques are called decision tree induction. It is difficult to realize a decision tree is when it is to be used for predicting data that has not yet been seen. When using a training set it is common for the generated decision tree to just fit the training data; new data will be incorrectly classified when using the generated decision tree. If the training set is classified completely, the generated decision tree will most likely overfit the data and new items will be misclassified. If stopping the generation somewhere before completion, the data could be misclassified when one more step would have correctly classified the data. The question is then how to manage setting up the decision tree without overfitting or underfitting the training data. Several techniques will be described later, but no perfect technique has been found.

2.6.3.2 Decision Tree Induction

The goal is decision tree induction is to correctly classify the data by the path to the leaves of the generated tree. There are basically two types of algorithms: split off groups of data that are as large as possible or divide into multiple groups of data as close to half as possible. Both of these are applied recursively down the tree until some stopping criteria [Decision Tree, 2007].
2.6.3.2.1 Gini Index

The Gini Index technique is of the type that splits off groups of data that are as large as possible. At each node, this is done by the measure of purity of a class of data relative to all of the classes of data at the node. Purity is measured as follows [Andriyashin, 1986]: given classes $a, b, c, d$, given node $t$, then let the probability that an observation in node $t$ falls in class $a$ is $p(a|t)$, then the probability that an observation in node $t$ not falling into class $a$ is $1 - p(a|t)$. Thus, for node $t$, the sample variable estimate of class $a$ is $p(a|t)(1 - p(a|t))$. Therefore, the total variance for node $t$ is:

$$p(a|t)(1 - p(a|t)) + p(b|t)(1 - p(b|t)) + p(c|t)(1 - p(c|t)) + p(d|t)(1 - p(d|t))$$

$$= [p(a|t) + p(b|t) + p(c|t) + p(d|t)] - \{[p(a|t)]^2 + [p(b|t)]^2 + [p(c|t)]^2 + [p(d|t)]^2\}$$

$$= 1 - \{[p(a|t)]^2 + [p(b|t)]^2 + [p(c|t)]^2 + [p(d|t)]^2\}$$

[Andriyashin, 1986]

In general, given $J$ classes in the training set, the impurity at node $t$ is measured by the Gini Index:

$$i(t) = \sum_{j=1}^{J} p(j \mid t)(1 - p(j \mid t)) = 1 - \sum_{j=1}^{J} p^2(j \mid t)$$

[Andriyashin, 1986] (45)

Denote an arbitrary split $s$, given a node $t$, with two child nodes $t_L$ and $t_R$. A fraction $p_L$ of data from $t$ falls to the left child node $t_L$ and $p_R = 1 - p_L$. The impurity of the right sub-tree $t_R$ is $i(t_R)$ and the impurity of the left tree $t_L$ is $i(t_L)$. The total impurity of $t_L$ and $t_R$ is $p_Ri(t_R) + p_Li(t_L)$. For a particular split $s$ and node $t$, the impurity is reduced by

$$\Delta i(s, t) = i(t) - p_Li(t_L) - p_Ri(t_R)$$

[Andriyashin, 2005] (46)
Thus, recursively all possible splits are tested to find the one with the largest reduction of impurity.

2.6.3.2.2 Gini Diversity Index

The Gini Diversity Index [Grajski, 1986] guides the maximal class separation by reducing misclassification of cases at node $t$. The impurity function estimates the probability of misclassification in node $t$. This is done by randomly selecting a case in $t$ and assigning it class label $i$ with probability $p(i|t)$ of being correct. The probably that the case is actually labeled $j$ is $p(j|t)$. The probability of misclassification is estimated to be:

$$i(t) = \sum_{j \neq i} p(t|t) p(j|t)$$

(47)

[Grajski, 1986]

Again, this equation is maximized to find the best split:

$$\Delta i(s,t) = i(t) - p_L i(t_L) - p_R i(t_R)$$

(48)

[Grajski, 1986]

2.6.3.2.3 Information Gain

The concept of information gain was in the context of decision trees was popularized by Quinlan [1986]. In this case, division of nodes is terminated when there is no significant decrease in the measure of impurity in the possible children. The node then becomes a terminal node with the majority class represented by the node the classification at that node. In other words, class $w$ at terminal node $t$ which maximizes the conditional probability $p(w|t)$. Here, for $k$ classes, the impurity is measured by
\[ i(t) = -\sum_{j=1}^{k} p(w_j \mid t) \log p(w_j \mid t) \]  
\text{[Quinlan, 1986]}  

Therefore, the best split is the maximum of the following for all possible splits

\[ \Delta i(s, t) = i(t) - p_L i(t_L) - p_R i(t_R) \]  
\text{[Quinlan, 1986]}

### 2.6.3.2.4 Twoing Rule

The twoing rule basically strikes a balance between purity and creating roughly equal-sized nodes [CART slides]. The measure of purity here is as follows: at a node \( S \), with children \( S_1 \) and \( S_2 \), with \( J \) classes,

\[ S_i(s) = \{ j : p(j \mid t_L) \geq p(j \mid t_R) \} \]

\[ \max_{S_1} \Delta i(s, i, S_i) = \frac{p_L p_R}{4} \left[ \sum_{j=1}^{J} |p(j \mid t_L) - p(j \mid t_R)| \right]^2 \]  
\text{[Andriyashin, 2005]}

This equation is then maximized to determine the split.

### 2.6.3.3 Optimal Termination

As mentioned before, one of the difficult parts of building decision trees is deciding when to stop the process. Typically a training data set is used to build the tree which is only a representation of all possible data cases. If the decision tree is induced until all leaves are purely one class, it is likely that future data will be misclassified because all dimensions used to determine that classification might not be the same in additional data. On the other hand, if induction is terminated at some arbitrary point before the leaves, it
is quite possible that a small number of additional splits would have led to a more accurate tree.

In the cases above, $\Delta i(s,t)$ is used to determine the split. The obvious place to stop is when $\Delta i(s,t) = 0$, or in other words the leave nodes are maximally pure, which has already been discussed as not desired. Another stopping rule needs to be created such as

$$\Delta i(s,t) < \beta$$

[Andriyashin, 2005]

where $\beta$ is some threshold value. If all possible splits are calculated and $\max \Delta i(s,t) < \beta$ then there is no split. It can be shown that impurity calculations are not monotonic while building the tree [Andriyashin, 2005]. This makes determining $\beta$ difficult since it may determined that no split is done at this node while if a less than desired split was done, the impurity measure at the child node would reveal a high information gain.

Another option when deciding when to terminate decision tree induction is to stop splitting when the children would have too few cases:

$$N(t) > \tilde{N}$$

[Andriyashin, 2005]

where $N(t)$ is the number of cases at node $t$ and $\tilde{N}$ is the threshold. While the number of cases at a node is always monotonic by definition, selecting this value is challenging like selecting $\beta$. Often, simulation is used to determine these values. This leads to values that are not globally optimal, only values that suit the data at hand [Andriyashin, 2005].
2.6.3.4 Pruning

Another way to induce a decision tree is to build the tree completely, overfitting the data, and then create a measure to determine when undesired parts of the tree should be pruned. In other words, sub-trees are replaced by leaves based on different rules described here.

2.6.3.4.1 Cost-Complexity Pruning

Breiman [1984] proposed a two-stage process for pruning decision trees. The first step entails generating trees $T_0, T_1, \ldots, T_k$ where $T_0$ is the original decision tree and $T_{i+1}$ is arrived at by replacing one or subtrees of $T_i$ with leaves until $T_k$ is only one node. The cost-complexity of a tree $T$ is defined as

$$\frac{E}{N} + \alpha \times L(T)$$

[Breiman, 1984]

where $E$ is the number of misclassified cases, $L(T)$ is the number of leaves in $T$, and $\alpha$ is some parameter. Suppose some subtree $S$ of $T$ is replaced by the best possible leaf. The new tree would misclassify $M$ more cases but would contain $L(S) - 1$ fewer leaves. The new tree would have the same cost-complexity as $T$ if

$$\alpha = \frac{M}{N \times (L(S) - 1)}$$

[Breiman, 1984]

$T_{i+1}$ is produced by finding the minimum $\alpha$ and replacing the subtree or subtrees with the minimum $\alpha$ with their respective best leaves [Breiman, 1984].
For an example [Breiman, 1984], consider the subtree below, where the number in the parentheses is the number of cases covered by that leaf:

\[
\begin{align*}
T4U \text{ measured } &= t: \text{ negative (1918)} \\
T4U \text{ measured } &= f: \\
| & \quad \text{age} > 43.5: \text{ negative (58)} \\
| & \quad \text{age} < 43.5: \\
| | & \quad \text{query hypothyroid } = f: \text{ negative (41)} \\
| | & \quad \text{query hypothyroid } = t: \text{ secondary hypothyroid (1)}
\end{align*}
\]

[Breiman, 1984]

The majority of cases at the leaves of this tree are negative with one non-negative case.

If this subtree was replaced by the leaf negative, the number of cases misclassified \( M \) is 1 out of 2018 cases, giving a value for \( \alpha \) of 0.00013 for which the cost-complexity of the original and new trees would be equal [Breiman, 1984].

The second stage of Breiman’s [1984] model throws out the training set of data and uses another set of test data from which to determine the reliability of the possible \( T_0, \ldots, T_k \) decision trees. For a test set containing \( N' \) cases, \( E' \) is the minimum number of errors observed with a given \( T_i \), the standard error of \( E' \) is given by

\[
se(E') = \sqrt{\frac{E' \times (N' - E')}{{N'} \times N'}}
\]

[Breiman, 1984]

The final tree \( T_i \) selected is the smallest tree whose number of errors in the test set does not exceed \( E' + se(E') \) [Breiman, 1984].
2.6.3.4.2 Reduced Error Pruning

Breiman [1984] proposed another pruning method as follows. Given a test set independent of the training set, the new number of misclassifications in every non-leaf subtree is calculated. Subtrees of the generated tree are replaced by one of its best possible leaves if the new number of misclassifications is equal to or fewer than the original number of errors, the subtree is replaced by the best leaf. The advantage of this method is that its rationale is clearer than the previously described method.

2.6.3.5 Decision Trees Algorithm

Generation of a decision tree is composed of two major steps: tree induction and either optimal termination of induction or pruning of subtrees after it is completely generated. Four methods of tree induction have been illustrated, two methods of termination have been described, and two types of pruning have been highlighted. One possible choice of algorithm follows.

Given a training set, where each case represents the correct classification and dimensions of that classification. This training set is represented in its entirety at the root of the tree, the recursive algorithm follows:

1. At node $t$, the impurity of this node is calculated by equation 45.
2. The node is split into the children nodes $t_R$ and $t_L$ chosen by looking at all possible splits. For a given split $s$, the impurity is calculated to be reduced by equation 46. Thus, $t_R$ and $t_L$ are chosen based on the largest reduction in purity.
3. Loop back to step 1 until all leaves are pure, that is the leaves contain only one class.
4. Given a test set, where the data are independent of the training set, the number of incorrect classifications is determined.

5. For a given non-leaf subtree, if the subtree were replaced with a leaf representing the class with the highest number of examples, the number of misclassifications in the test set is again calculated.

6. If the number of misclassifications for this new tree is equal to or small than the number of misclassifications in the original tree, replace the subtree with the proposed leaf.

7. Loop from 5 until there are no subtrees that can be replaced by subtrees with the same or smaller number of misclassifications.

2.6.3.6 CART

When the data is continuous, there is a modification made in the calculation for splitting a node into its children. The CART [Andriyashin, 2005] model uses squared residual minimization. The sum of child nodes’ variances is calculated. The split with the minimum sum of variances is chosen: \( \text{min}[\text{Var}(Y_{\text{left}}) + \text{Var}(Y_{\text{right}})] \) where

\[
\text{Var}(Y^d) = \sum_{i=1}^{N_d} (Y^d_i - \frac{1}{N_d} \sum_{i=1}^{N_d} Y^d_i)^2
\]

[Andriyashin, 2005]

where \( d \) is \{left, right\}, \( Y_i \) is the response, and \( N_i \) is the node size.

2.7 Comparison of Techniques

The previous section described the techniques in detail. This section compares the
techniques discussed in this thesis, giving advantages and disadvantages of each.

2.7.1 Decision Trees

The major advantage of decision trees is that they are simple to understand and interpret. Disadvantages of decision trees: their use in prediction is not so straightforward as they employ techniques that are either experimentally determined, in the case of optimal termination, or based on models with poorly understood superiority, such as in Breiman’s [1984] cost-complexity model. These problems lead to solutions that are less than optimal [Breiman, 1984]. Decision trees must have an extensive training set of data, and in some cases a test set of data. The tree needs to be re-generated when re-training is desired, when new cases should be included in the classification decision. The computational complexity of generating decision trees is high; growing the tree and pruning the tree can each take many passes over the tree and over the data set. Training techniques lead to trees that are not optimal because of overfitting or underfitting problems.

2.7.2 Neural Networks

The advantage to a neural network is that they can be quite accurate when properly set up. If a large amount of computational resources are available, much of the work generating a neural network can be done in parallel. This leads to leads to nearly linear computational complexity in the size of the data set. The disadvantage of a neural network the mathematical foundation and path taken for the weights generated for the neural network is virtually incomprehensible; it requires enormous analysis to understand
the parameters of the system. Some call neural networks a black-box system. Another
disadvantage is that the training method and equations used to generate the system are
critical to the accuracy of the system. The designer must very carefully select the
training methodology, equations, and thresholds if applicable to the chosen methods.
Thus, the accuracy of the system by definition is questionable. The end-user of such a
system will not have an intuitive basis to trust the resulting model. Other issues with
neural networks include extensive training, the fact that use of the training data to often
leads to model that are overfit or possibly underfit, and that new data points require
re-training.

2.7.3 Rule learning

The advantage of rule learning is that it is understandable by a typical user. Also,
IREP works well with noisy data [Fürnkranz, 1994]. Unfortunately, rule learning has
the disadvantage that it does not scale well with the size of the data: \( O(n^4) \) for REP,
\( O(n^2 \log^2 n) \) for IREP, and \( O(n \log^2 n) \) for RIPPER. Similar to other learning strategies,
training requires a large data set for relatively accurate results, rule learning has many
overfit and underfit problems in its design, and re-training is required with new data.
Further, rule learning often employs a greedy strategy leading to less than optimal results.

2.7.4 Regression

The advantage of regression is that it has low computational complexity. Though,
least squares estimation for linear models is notoriously non-robust to outliers. If the
distribution of the outliers is skewed, the estimates can be biased. Outliers can cause the
least squares estimates to be extremely off. Additionally, regression requires a large history for training and re-training is required with new data.

2.7.5 Q-learning

The advantage of Q-learning is that it fits naturally to problems where there are a small number of goals. The disadvantage is that this method requires a long time of training to converge and everything has to be relearned when there is a change to the system.

2.7.6 Moving Average

Moving average is easy to understand and it has a low computational complexity. The downside is that it the moving average lags behind the current data. Since it is averaging the last $k$-pieces of data, it is less accurate for new data.

2.7.7 Stacked Generalization

Stacked generalization combines the outputs of different classifiers to produce a higher quality output; it is not a classifier in itself. Being related to the other classifiers, it consequently has the same advantages and disadvantages, possibly amplified in either direction giving the selection and combination of classifiers.

2.7.8 Solution to Secretary Problem

The advantages of the solution to the Secretary Problem are that it has a low computational complexity and is straight forward for the average person to understand,
remember, and use. Though, it gives poor results when the temporal data does not increase or decrease as expected.

2.7.9 Algorithm Tradeoffs

This is a chart of some of the tradeoffs between the algorithms reviewed:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Complexity</th>
<th>Design Difficulty</th>
<th>Use Difficulty</th>
<th>Training</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule Learning</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>X</td>
<td>Moderate</td>
</tr>
<tr>
<td>Q-Learning</td>
<td>Moderate</td>
<td>Moderate</td>
<td>Moderate</td>
<td>X</td>
<td>Moderate</td>
</tr>
<tr>
<td>Moving Average</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>Regression</td>
<td>Moderate</td>
<td>Low</td>
<td>Low</td>
<td>X</td>
<td>Moderate</td>
</tr>
<tr>
<td>Neural Networks</td>
<td>Moderate</td>
<td>High</td>
<td>High</td>
<td>X</td>
<td>High</td>
</tr>
<tr>
<td>Decision Trees</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>X</td>
<td>Moderate</td>
</tr>
<tr>
<td>Arbitrary Loss</td>
<td>Low</td>
<td>Moderate</td>
<td>Low</td>
<td></td>
<td>Moderate</td>
</tr>
<tr>
<td>Minimize Rank</td>
<td>Low</td>
<td>Moderate</td>
<td>Moderate</td>
<td></td>
<td>Moderate</td>
</tr>
<tr>
<td>Secretary</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td></td>
<td>Moderate</td>
</tr>
</tbody>
</table>

2.7.10 Algorithm Selection

One consideration when selecting the algorithm to use to apply to airline ticket data was realized after studying the literature: consumers prefer algorithms that are user-friendly and make (intuitive) sense. This immediately leaves out neural networks and Q-learning. Regression was also discarded because the influences from many sources of data will not be apparent in a trend line against ticket data on a particular flight. Moving average was not used because rapid changes in data are going to be the interesting points and moving average typically lags behind the most recent data since it is averaging the previous k-points. Decision trees are very user-friendly and rule learning is straight forward to follow. The problem with these two, from a user perspective, is that the systems need to retrain at each price observation and the tree or rules may change without warning. An unanticipated change may confuse the
consumer. This may make some consumers untrusting of the model. The solution to the Secretary Problem, an optimal stopping problem, is ideal for picking up points where values suddenly change to extremes. Upon observation of ticket prices, sudden changes are apparent. The variations to the Secretary Problem are unnecessarily complicated in the case of only one choice. Thus, the solution to the Secretary Problem was chosen to take advantage of the sudden changes in price.
3 Technology

This chapter reviews the technology used for this thesis.

3.1 Introduction

The World Wide Web provides an excellent method to autonomously gather airline ticket pricing data. It is almost always available and programs (known as bots or spiders) are written to parse web pages without user intervention. Often, many pages on one site are all similar and pulled from a database into a template. This inherent structure allows a program to systematically gather the desired information. One downfall to this is that a change to the template structure requires the program to adapt to the new structure.

For this investigation, there are three websites categories considered: online travel agencies, meta, and airline. The three major discount sites are Expedia, Orbitz, and Travelocity. Meta sites are ones that scour dozens to hundreds of airlines. Meta sites are not interesting for this study since they cannot directly be parsed; they only link to airline websites instead of displaying the final price and selling tickets themselves. While navigating travel sites, the price of interest is the price including taxes and fees, such as service fees. The following six sites were targeted for navigation: Delta, Continental, NWA, Orbitz, Expedia, and Travelocity. NWA and Travelocity do not respond to web browser mimicking, so the author is led to believe that activities occur on the page other than the action moving between pages. Orbitz’s results page is compressed by an unknown algorithm. Expedia’s results page is optionally compressed.
For a short time, Continental’s results pages were encrypted, but luckily they were returned to unencrypted. Due to these problems, the final selection of sites is Delta, Continental, and Expedia.

3.2 Protocols

The technology for retrieving information over the World Wide Web is the HTTP protocol. This protocol defines standard formats for information exchange between server and client. This thesis uses HTTP to communicate with select websites that offer airline tickets to gather pricing information over time. There are two parts to the HTTP protocol: the client request and the server response. Common fields are explained here following an example:

**Client request**

GET /index.html HTTP/1.1

Host: www.example.com

**Server response**

HTTP/1.1 200 OK

Date: Mon, 23 May 2005 22:38:34 GMT

Server: Apache/1.3.27 (Unix) (Red-Hat/Linux)


Content-Length: 438

Connection: close

Content-Type: text/html
Definitions

1. GET: request the resource
2. POST: submits data along with requesting the resource
3. Accept: file types to accept from the request
4. Accept-Encoding: the format of the returned data
5. Accept-Language: the natural language preferred in response
6. Host: the host being requested
7. Referer[sic]: the URI from which the requested URI was obtained
8. Content-length: the number of bytes for the message-body
9. Content-Type: the format of the content
10. Connection: specifies whether to keep the connection open or to close it

[HTTP, 2006; Fielding, 1999].

Java is used to employ the HTTP protocol in the exchange of web page requests, web page receipt, and parsing the returned web pages. Here, airline tickets prices are gathered from the chosen websites. Java uses a concept of sockets to specify the port and network protocol used for machine contact, such as TPC. For this project, the behavior of a web browser is mimicked by exchanging HTTP packets. The packets are gathered with Ethereal (Ethereal, Inc. - http://www.etheral.com) and studied. Additionally the SMTP protocol is employed to alert the author of failures.
Next is a sample communication with www.example.com for sending a mail from sender@mydomain.com to friend@example.com, where S signifies a message from the server to the client and C signifies a message from the client to the server:

S: 220 www.example.com ESMTP Postfix

C: HELO mydomain.com

S: 250 Hello mydomain.com

C: MAIL FROM:<sender@mydomain.com>

S: 250 Ok

C: RCPT TO:<friend@example.com>

S: 250 Ok

C: DATA

S: 354 End data with <CR><LF>.<CR><LF>

C: Subject: test message

C: From: sender@mydomain.com

C: To: friend@example.com

C:

C: Hello World

C: .

S: 250 Ok: queued as 12345

C: QUIT

S: 221 Bye

[Postel, 1982; SMTP, 2006].
The important parts are as follows: “MAIL FROM” is the sender of the email, “RCPT TO” is the recipient of the email, “DATA” begins the email, and a line with only a period terminates the email. It is useful to send automated emails so that one does not have to continuously monitor the system and yet immediately be alerted when there is a problem.

3.3 Compression

Historically, the most time consuming part of a client-server communication is the transfer of a file over the network. Therefore, compression is used to limit the size of files to be transferred. Orbitz’s search results are always compressed. Expedia’s results are optionally compressed based on a specified HTTP string with the desired format. The three standard formats are gzip, deflate, and compress [Fielding, 1999]. Gzip is an encoding format produced by the file compression program “gzip” (GNU zip) as described in RFC 1952 [Deutsch, 1996c]. This format is a Lempel-Ziv coding (LZ77) with a 32 bit CRC [Deutsch, 1996c]. Compress is an encoding format produced by the common UNIX file compression program “compress.” This format is an adaptive Lempel-Ziv-Welch coding (LZW) [Deutsch, 1996b]. Deflate is the “zlib” format defined in RFC 1950 in combination with the “deflate” compression mechanism described in RFC 1951 [Deutsch, 1999a; Deutsch, 1999b]. The decompression algorithms included with Java were unable to handle the formats from Orbitz and Expedia. Therefore, Orbitz is dropped and compression is turned off from Expedia packets.
4 Approach

The problem was introduced in chapter 1, background and theory for the problem was discussed in chapter 2, and the technology was discussed in chapter 3. This chapter contains the experimental design, the data, and the following analysis.

4.1 Experimental Design

This section describes the experimental design of this thesis. The Secretary Algorithm, in summary, tracks the minimum ticket price value seen so far until $1/e$, or 36.8%, of the choices have been seen and then selects the next value that is below the minimum value recorded before the boundary. If all ticket prices are exhausted and there was not a lower ticket price seen after the boundary, then the last price is selected.

In another form, the step by step algorithm follows:

1. For each site, for each route, loop
2. Determine $N$, the number of ticket prices to examine
3. Examine the first $\left\lceil N/e \right\rceil$ ticket prices and record the minimum price
4. Examine the remaining ticket prices and determine lowest ticket price below the minimum price before the boundary
5. If none were found lower than the price before the boundary, the result is the last price.

Six routes were studied: BOS-SFO, CVG-BZN, CVG-SFO, DAY-SFO, LAX-BOS, and SEA-IAD, and three websites were parsed: Delta, Continental, and Expedia. CVG-BZN for each website was gathered as a weekend trip (Friday, May 26, 2006 to Sunday, May 28, 2006) while the others were gathered as a week long trip (Monday, May
22, 2006 to Friday, May 26, 2006). These dates were chosen arbitrarily; they are representative of an average search of a period without major holidays. The data gathering began on April 12, 2006. Thus, data were gathered for CVG-BZN over 44 days and data for the others were gathered over 39 days. Data were gathered every 30 minutes. The data were gathered sequentially for each website and route. There is approximately a five-minute gap between the data gathering of each website because of the time it takes to gather the data for the six routes on the preceding website. The data are not consistently gathered for the following reasons: power outage at the gathering machine’s location, remote website outage, bugs in the parser, and the website format being changed during gathering. For ease of analysis, the times were rounded to the preceding half an hour. Further, to study the Secretary Algorithm in a realistic setting, at each data point a customer is simulated to arrive desiring to purchase a ticket. The Secretary Algorithm is run for each of these customers and the ticket costs are recorded.

The Secretary Algorithm previously described is too strict to be practically useful. The default of accepting the final ticket price causes just over half of the site(route combinations to lose customers money. Additional conditions based on examination of the data and anecdotal experiences are imposed as follows:

1. If a customer arrives or considers a ticket within the last 6 days before departure, make sure the customer is not currently at a spike and then purchase the ticket.
2. If the price rises by 8%, and it is not a spike, purchase the ticket. Otherwise, allow the search to continue for the price that is lower than the best price before the boundary.
(The threshold of 8% was chosen experimentally for this set of data by looking at values near the standard scientific values of 5% and 10%.)

In the data trend study, the following are observed: positive spikes, positive jumps, negative spikes, and negative jumps, where a spike is defined as a change in price for four time periods or less and a jump is defined as a change in price for over four time periods, where a time period is half an hour. These changes are recorded and graphs are generated.

Hypotheses are tested as follows:
I. The results of the Secretary Problem being applied to the data will be reviewed to see if consumers save money using this method.
II. The general trend of the data will be observed, including the monotonicity of the data.
III. The data will be reviewed for the worst period to purchase an airline ticket; it is believed the last week before departure is the worst time to buy a ticket.
IV. The data will be reviewed for volatility; in particular, the frequency of price changes, the rate of change in that frequency, and the occurrence of a sudden change in price at a multiple of a week from departure.
V. The data will be reviewed for the most beneficial lead time to begin looking for an airline ticket.
VI. The tendency for there to be a small number of unique prices for a given flight will be reviewed from the data.
4.2 Data & Results

This section shows the data and following analysis of the data.

4.2.1 Secretary & Algorithm Data & Results

![Figure 13](image)

Figure 13: Secretary Algorithm on BOS-SFO route combined over each website

Figure 13 shows the data gathered from the route Boston (BOS) and San Francisco (SFO) as well as the Secretary Algorithm applied on the entire range of data. (See appendix 7.1 for the rest of the routes.) This one was chosen because it shows an example of the Secretary Algorithm succeeding, in the case of Delta and Continental, and an example of the Secretary Algorithm failing to find a discount price, in the case of Expedia. Generally, the difference between these two series of data is that the first two have a drop in price before departure while Expedia never decreases below the minimum value before the boundary. It is useful to point out that the application of this algorithm
does not guarantee the global minimum in the data; it returns a price that is lower than the initial price considered, or the final point if a lower one does not exist. This graph of the three data series has the secretary algorithm applied independently on one graph.

The data are studied to provide insights into an enhanced algorithm that could provide a result better than the final point the Secretary Algorithm often returns. An explanation of how the data is examined follows. The data from figure 13 are from the viewpoint of a customer whom arrives at the beginning of sampling. It is more appropriate to assume that a customer can first view ticket prices at any point before departure. Thus, the algorithm is run as if a customer begins viewing the data at each sample point. For an example, see figure 14, where the number 1 represents a customer beginning their airline ticket search at about midnight on April 15. The customer observes ticket prices but does not consider purchasing them until 36.8% of the remaining anticipated ticket price observations have passed which is on April 29 at 7am pointed by the number 2 in this example. The purchase price point for this customer is recommended at number 3, the first ticket lower than all of those seen so far, which is $499.20. The graphs of customers appearing at each point are seen in figures 15 and 16, where the pink line indicates the purchase price from the Secretary Algorithm as described. The pink line does not indicate the date the ticket was purchased; this can be determined by the next airline data point that is at the same price as the pink data point. Figures 15 and 16 are annotated with a letter indicating the group of customers arriving on a given date and the purchase price for the group of customers. For example, in figure 15, section B covers customers that arrive from April 14 through April 25,
approximately 500 customers, and purchases a ticket on the night of May 3 for $499.20. Note that in Figure 16, section C represents the final two sets of customers purchasing the last ticket available when using this algorithm. The images were annotated in this manner to exemplify the algorithm in action over many different possible customers, and this was done to visualize the computations. In an actual airline purchase scenario there might be many customers that decide to begin their airline ticket search at a certain time, say at noon or 1:00 p.m. during their lunch break, and in other time periods, there might be few or no new customers beginning their ticket search, say at 3:00 a.m. on a Tuesday. The markers are to illustrate each customer beginning their airline ticket search and what would happen over the course of the search if the search was automated and all ticket price data were available.

![Secretary - DAY-SFO Continental](image)

Figure 14: Annotated example for Secretary Algorithm with a single customer
Figure 15: Successful Secretary Algorithm on DAY-SFO route from Continental

Figure 16: Moderately successful secretary algorithm on SEA-IAD route from Delta
Figure 15 is representative of just under half of the results. The algorithm is successful if the customer arrives up to 2.5 weeks before departure where the ticket is purchased up to 1.5 weeks before departure. The graph shows money saved for 621 out of 1351 customers with an average savings of $108.14 when savings are possible.

Figure 16 is representative of some additional cases. The algorithm succeeds for short periods because of a local drop in price. For the rest of the cases (illustrated by the lack of a pink data point), the Secretary Algorithm selects the last sample point to purchase the ticket because the price consistently rises. (See appendix 7.2 for the rest of the graphs of Secretary Algorithm results.) It is first studied in this manner to observe when the Secretary Algorithm is successful so that this information can be used to enhance the algorithm.

Table 1 shows the number of contiguous days before plane departure the Secretary Algorithm saves money. If a customer begins to look for an airline ticket on the given number of days before departure, he or she can successfully run the algorithm. Table 2 shows the number of customers (and percentage of customers) which saved money with Secretary Algorithm excluding the cases where the algorithm only gives the final price. Table 3 shows the average savings per customer when savings are possible. Table 4 shows the average savings per customer over all of the data.

<table>
<thead>
<tr>
<th></th>
<th>Delta</th>
<th>Continental</th>
<th>Expedia</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOS-SFO</td>
<td>25</td>
<td>19</td>
<td>-</td>
</tr>
<tr>
<td>CVG-BZN</td>
<td>-</td>
<td>27</td>
<td>-</td>
</tr>
<tr>
<td>CVG-SFO</td>
<td>-</td>
<td>19</td>
<td>38</td>
</tr>
<tr>
<td>DAY-SFO</td>
<td>19</td>
<td>19</td>
<td>-</td>
</tr>
<tr>
<td>LAX-BOS</td>
<td>-</td>
<td>19</td>
<td>-</td>
</tr>
<tr>
<td>SEA-IAD</td>
<td>34</td>
<td>19</td>
<td>25</td>
</tr>
</tbody>
</table>
Table 1: Customer must begin searching for a ticket the above number of days before plane departure for the Secretary Algorithm to save money

<table>
<thead>
<tr>
<th>Route</th>
<th>Delta</th>
<th>Continental</th>
<th>Expedia</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOS-SFO</td>
<td>39.61%</td>
<td>45.37%</td>
<td>0.15%</td>
</tr>
<tr>
<td>CVG-BZN</td>
<td>3.79%</td>
<td>17.44%</td>
<td>0.96%</td>
</tr>
<tr>
<td>CVG-SFO</td>
<td>0.07%</td>
<td>45.62%</td>
<td>13.87%</td>
</tr>
<tr>
<td>DAY-SFO</td>
<td>44.59%</td>
<td>45.30%</td>
<td>3.04%</td>
</tr>
<tr>
<td>LAX-BOS</td>
<td>5.53%</td>
<td>45.89%</td>
<td>0.15%</td>
</tr>
<tr>
<td>SEA-IAD</td>
<td>21.63%</td>
<td>45.50%</td>
<td>41.10%</td>
</tr>
</tbody>
</table>

Table 2: Number of customers (and percentage) which saved money with the Secretary Algorithm

<table>
<thead>
<tr>
<th>Route</th>
<th>Delta</th>
<th>Continental</th>
<th>Expedia</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOS-SFO</td>
<td>$205.16</td>
<td>$107.48</td>
<td>$105.85</td>
</tr>
<tr>
<td>CVG-BZN</td>
<td>$258.26</td>
<td>$191.40</td>
<td>$91.60</td>
</tr>
<tr>
<td>CVG-SFO</td>
<td>$0.00</td>
<td>$108.27</td>
<td>$119.92</td>
</tr>
<tr>
<td>DAY-SFO</td>
<td>$273.08</td>
<td>$108.14</td>
<td>$43.54</td>
</tr>
<tr>
<td>LAX-BOS</td>
<td>$46.95</td>
<td>$108.52</td>
<td>$33.25</td>
</tr>
<tr>
<td>SEA-IAD</td>
<td>$0.16</td>
<td>$108.41</td>
<td>$231.31</td>
</tr>
</tbody>
</table>

Table 3: Average savings per customer when savings are possible

<table>
<thead>
<tr>
<th>Route</th>
<th>Delta</th>
<th>Continental</th>
<th>Expedia</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOS-SFO</td>
<td>-$114.55</td>
<td>-$100.43</td>
<td>-$217.13</td>
</tr>
<tr>
<td>CVG-BZN</td>
<td>-$493.35</td>
<td>-$250.90</td>
<td>-$934.06</td>
</tr>
<tr>
<td>CVG-SFO</td>
<td>-$630.25</td>
<td>-$99.93</td>
<td>-$200.20</td>
</tr>
<tr>
<td>DAY-SFO</td>
<td>-$134.66</td>
<td>-$100.29</td>
<td>-$205.59</td>
</tr>
<tr>
<td>LAX-BOS</td>
<td>-$215.05</td>
<td>-$99.22</td>
<td>-$210.60</td>
</tr>
<tr>
<td>SEA-IAD</td>
<td>-$175.23</td>
<td>-$100.53</td>
<td>-$218.66</td>
</tr>
</tbody>
</table>

Table 4: Average loss per customer when using the theoretical Secretary Algorithm

In the half of the cases that led the Secretary Algorithm to fail the program would have originally purchased the final ticket available and the final ticket available is quite expensive. To prevent such losses, the Secretary Algorithm is adapted to immediately purchase a ticket if the current price being is over some threshold, purchase a ticket with the basic Secretary Algorithm when the conditions are met, or when in the last six days before departure simply purchase the ticket. Additionally, in each case four time steps
(two hours) are passed to test for spikes in the price so that spikes may be ignored. This leads to the average savings and losses shown in table 5. The cumulative average is a loss of $4.18 per customer. Table 5 shows that the airline Continental is most appropriate to the adapted algorithm, and this is because of the drop in price common to that airline. (See appendix 7.3 for all algorithm graphs.) Table 6 shows the dollar amounts lost or gained for each airline and route pair. When money is saved, $456,407.02 is saved, but when money is lost, $672,771.53 is lost, for a total loss of $216,364.51.

![Algorithm - DAY-SFO Delta](image)

Figure 17: Algorithm in action on DAY-SFO on Delta
<table>
<thead>
<tr>
<th></th>
<th>Delta</th>
<th>Continental</th>
<th>Expedia</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOS-SFO</td>
<td>-22203.19</td>
<td>-60376.10</td>
<td>54300.20</td>
<td>-28279.09</td>
</tr>
<tr>
<td>CVG-BZN</td>
<td>98853.54</td>
<td>-24194.20</td>
<td>71633.61</td>
<td>146292.95</td>
</tr>
<tr>
<td>CVG-SFO</td>
<td>86856.92</td>
<td>-60761.10</td>
<td>139506.29</td>
<td>165602.11</td>
</tr>
<tr>
<td>DAY-SFO</td>
<td>-57863.38</td>
<td>-60630.10</td>
<td>78168.83</td>
<td>-40324.65</td>
</tr>
<tr>
<td>LAX-BOS</td>
<td>54742.02</td>
<td>-60549.90</td>
<td>59823.24</td>
<td>54015.36</td>
</tr>
<tr>
<td>SEA-IAD</td>
<td>28886.88</td>
<td>-59585.80</td>
<td>-50243.25</td>
<td>-80942.17</td>
</tr>
<tr>
<td>Sum</td>
<td>189272.79</td>
<td>-326097.20</td>
<td>353188.92</td>
<td>216364.51</td>
</tr>
</tbody>
</table>

Table 5: Average savings and losses per customer for the adapted algorithm

<table>
<thead>
<tr>
<th></th>
<th>Delta</th>
<th>Continental</th>
<th>Expedia</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOS-SFO</td>
<td>$16.53</td>
<td>$45.06</td>
<td>-$40.59</td>
<td></td>
</tr>
<tr>
<td>CVG-BZN</td>
<td>-$64.65</td>
<td>$15.74</td>
<td>-$48.96</td>
<td></td>
</tr>
<tr>
<td>CVG-SFO</td>
<td>-$65.06</td>
<td>$45.45</td>
<td>-$104.58</td>
<td></td>
</tr>
<tr>
<td>DAY-SFO</td>
<td>$43.21</td>
<td>$44.88</td>
<td>-$57.90</td>
<td></td>
</tr>
<tr>
<td>LAX-BOS</td>
<td>-$40.94</td>
<td>$45.25</td>
<td>-$45.49</td>
<td></td>
</tr>
<tr>
<td>SEA-IAD</td>
<td>-$21.77</td>
<td>$44.67</td>
<td>$37.75</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>-$13.61</td>
<td>$40.175</td>
<td>-$43.30</td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Savings and losses, dollar amounts

In summary, the Secretary Algorithm applied here saves just under half of the consumers money on their airline tickets.

4.2.2 Observations

One question that a customer might ask is at what time one should begin looking for an airline ticket (hypothesis V). Thus, the data is studied for the difference between the price of buying an airline ticket at the first price that is seen by the customer and the price the customer can buy a ticket using the adapted algorithm. Two notable patterns are seen in the graphs:

1. Three weeks and longer lead time is beneficial (see figure 18)
2. One to five weeks lead time is beneficial (see figure 19).
These are graphs of the difference between price the adapted algorithm determined to purchase the ticket and the first price the customer views. A negative number means the initial price is bigger than the algorithm purchase price.

Figure 18: Lead time study example – three weeks and longer lead time is beneficial
Table 7 shows an example of the bucket nature of the data, where bucket is a small range of prices for a set of seats. A five percent range in either direction is around the price indicated. The data has between three to six buckets per graph. (See appendix 7.5 for all bucket data.) Table 8 shows the average number of times the price changes per day per route and website. The cumulative average is 2.33 changes in price per day, with a range of 0 to 23 changes in a day. Table 9 contains the minimum prices for a given airline and route pair. Table 10 contains the average prices for a given airline and route pair.
<table>
<thead>
<tr>
<th>Price ($)</th>
<th># of points</th>
</tr>
</thead>
<tbody>
<tr>
<td>370.2</td>
<td>724</td>
</tr>
<tr>
<td>448.8</td>
<td>352</td>
</tr>
<tr>
<td>527.2</td>
<td>198</td>
</tr>
<tr>
<td>617.2</td>
<td>76</td>
</tr>
<tr>
<td>858.9</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 7: Buckets for BOS-SFO on Expedia

<table>
<thead>
<tr>
<th>Delta</th>
<th>Continental</th>
<th>Expedia</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOS-SFO</td>
<td>2.1</td>
<td>1.3</td>
</tr>
<tr>
<td>CVG-BZN</td>
<td>4.067</td>
<td>2.4</td>
</tr>
<tr>
<td>CVG-SFO</td>
<td>1.125</td>
<td>1.225</td>
</tr>
<tr>
<td>DAY-SFO</td>
<td>6.75</td>
<td>1.15</td>
</tr>
<tr>
<td>LAX-BOS</td>
<td>0.325</td>
<td>1.225</td>
</tr>
<tr>
<td>SEA-IAD</td>
<td>0.65</td>
<td>1.225</td>
</tr>
</tbody>
</table>

Table 8: Average number of times the price changes per day

<table>
<thead>
<tr>
<th>BOS-SFO</th>
<th>CVG-SFO</th>
<th>LAX-BOS</th>
<th>SEA-IAD</th>
<th>DAY-SFO</th>
<th>CVG-BZN</th>
</tr>
</thead>
<tbody>
<tr>
<td>449.19</td>
<td>569.60</td>
<td>358.60</td>
<td>644.69</td>
<td>504.20</td>
<td>453.70</td>
</tr>
<tr>
<td>359.80</td>
<td>359.80</td>
<td>359.80</td>
<td>359.80</td>
<td>359.80</td>
<td>389.80</td>
</tr>
<tr>
<td>370.20</td>
<td>303.70</td>
<td>363.60</td>
<td>266.20</td>
<td>266.20</td>
<td>362.60</td>
</tr>
<tr>
<td>393.06</td>
<td>411.03</td>
<td>360.67</td>
<td>423.56</td>
<td>376.73</td>
<td>402.03</td>
</tr>
</tbody>
</table>

Table 9: Minimum prices possible for each airline and route pair

<table>
<thead>
<tr>
<th>BOS-SFO</th>
<th>CVG-SFO</th>
<th>LAX-BOS</th>
<th>SEA-IAD</th>
<th>DAY-SFO</th>
<th>CVG-BZN</th>
</tr>
</thead>
<tbody>
<tr>
<td>591.45</td>
<td>693.67</td>
<td>445.11</td>
<td>759.93</td>
<td>846.14</td>
<td>924.64</td>
</tr>
<tr>
<td>548.61</td>
<td>548.71</td>
<td>548.67</td>
<td>548.64</td>
<td>549.02</td>
<td>552.60</td>
</tr>
<tr>
<td>431.52</td>
<td>506.29</td>
<td>418.32</td>
<td>514.90</td>
<td>390.09</td>
<td>771.49</td>
</tr>
<tr>
<td>523.86</td>
<td>582.89</td>
<td>470.70</td>
<td>607.82</td>
<td>595.08</td>
<td>749.58</td>
</tr>
</tbody>
</table>

Table 10: Average prices for each airline and route pair

Another observation is that at any data point it is possible for multiple airlines to have the same price on a ticket. In this case, the simple algorithm used is to take the
first in the list without prejudice. In this case, Expedia considered 509 tickets from Delta and two tickets from Continental out of 7821 tickets considered. Otherwise, the remaining 7309 tickets were from other airlines.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Delta</td>
<td>509</td>
</tr>
<tr>
<td>Continental</td>
<td>2</td>
</tr>
<tr>
<td>Alaska Airlines</td>
<td>5</td>
</tr>
<tr>
<td>American Airlines</td>
<td>1038</td>
</tr>
<tr>
<td>America West</td>
<td>223</td>
</tr>
<tr>
<td>Frontier Airlines</td>
<td>1</td>
</tr>
<tr>
<td>Midwest Airlines</td>
<td>23</td>
</tr>
<tr>
<td>Northwest</td>
<td>1248</td>
</tr>
<tr>
<td>United</td>
<td>3336</td>
</tr>
<tr>
<td>US Airways</td>
<td>1436</td>
</tr>
<tr>
<td><strong>Total Expedia Tickets</strong></td>
<td><strong>7821</strong></td>
</tr>
</tbody>
</table>

Table 11: Breakout of airlines tickets potentially purchased from Expedia

In the case where all tickets of the lowest price are considered, Expedia considers 1204 tickets from Delta and two tickets from Continental out of 18160 tickets considered over 7821 time periods considered. Otherwise, the remaining 16954 tickets considered were from other airlines.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Delta</td>
<td>1204</td>
</tr>
<tr>
<td>Continental</td>
<td>2</td>
</tr>
<tr>
<td>Alaska Airlines</td>
<td>214</td>
</tr>
<tr>
<td>American Airlines</td>
<td>2329</td>
</tr>
<tr>
<td>America West</td>
<td>472</td>
</tr>
<tr>
<td>Frontier Airlines</td>
<td>26</td>
</tr>
<tr>
<td>Midwest Airlines</td>
<td>23</td>
</tr>
<tr>
<td>Northwest</td>
<td>3191</td>
</tr>
<tr>
<td>United</td>
<td>8899</td>
</tr>
<tr>
<td>US Airways</td>
<td>1800</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>18160</strong></td>
</tr>
</tbody>
</table>

Table 12: Breakout of airlines tickets potentially purchased from Expedia including those at the same price
There are some overlaps in the ticket price consideration: tickets on Continental and Delta airlines are sold by Expedia. For the first ticket in the list taken without other consideration, there were 479 tickets chosen from Expedia’s inventory because they were cheaper than Delta’s offered price. When all tickets offered are considered regardless of they were purchased, there were 887 tickets offered by Expedia’s inventory for a Delta flight that were cheaper than Delta’s offered price.

Another study is comparing the purchase of the algorithm result to simply purchasing the first ticket seen. For some flights there is a savings of the algorithm price over the initial price, as in the case of Delta and BOS to SFO seen in Table 13, though the average loss of the initial price over the algorithm price is $35.23. This is compared to an average loss of $4.18 using the algorithm, seen in Table 14.

<table>
<thead>
<tr>
<th></th>
<th>BOS-SFO</th>
<th>CVG-BZN</th>
<th>CVG-SFO</th>
<th>DAY-SFO</th>
<th>LAX-BOS</th>
<th>SEA-IAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delta</td>
<td>-$160.01</td>
<td>-$290.00</td>
<td>$60.00</td>
<td>$210.00</td>
<td>$45.00</td>
<td>-$44.90</td>
</tr>
<tr>
<td>Continental</td>
<td>$56.00</td>
<td>$77.50</td>
<td>$56.00</td>
<td>$56.00</td>
<td>$56.00</td>
<td>$56.00</td>
</tr>
<tr>
<td>Expedia</td>
<td>$43.69</td>
<td>$65.10</td>
<td>$156.71</td>
<td>$52.00</td>
<td>$60.30</td>
<td>$78.70</td>
</tr>
</tbody>
</table>

Table 13: Ticket cost of algorithm ticket price minus the initial ticket price for the first customer

<table>
<thead>
<tr>
<th></th>
<th>BOS-SFO</th>
<th>CVG-BZN</th>
<th>CVG-SFO</th>
<th>DAY-SFO</th>
<th>LAX-BOS</th>
<th>SEA-IAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delta</td>
<td>-$0.61</td>
<td>$30.02</td>
<td>$62.45</td>
<td>-$55.08</td>
<td>$10.60</td>
<td>$30.94</td>
</tr>
<tr>
<td>Continental</td>
<td>-$59.28</td>
<td>$1.44</td>
<td>-$61.50</td>
<td>-$58.98</td>
<td>-$56.38</td>
<td>-$59.76</td>
</tr>
<tr>
<td>Expedia</td>
<td>$42.08</td>
<td>$46.14</td>
<td>$105.49</td>
<td>$51.98</td>
<td>$44.81</td>
<td>-$28.66</td>
</tr>
</tbody>
</table>

Table 14: Ticket cost of algorithm ticket price minus the initial ticket price average

Further, it is observed that there are uneven intervals created because there are missing data and this troublesome since a percentage of the data is used as a boundary point. The above analysis was done skipping and not considering the missing data. This was done because the tickets are ultimately not available for purchase directly by the
consumer for reasons outside of the consumer’s control. This causes some tickets to be purchased cheaper and others to be more expensive as discussed shortly.

It is interesting to consider the case when the consumer is unable to view ticket prices because of a problem on the consumer’s side of the system, such as an unavailable Internet connection failure, while the distributor is still in operation. Thus, for the purposes of determining the location of the 36.8% boundary, the offering ticket price could be estimated based on the available data. One solution could be to fill in the missing data by interpolating based on the end points of the available data. There are different techniques for interpolating the data including extending one or both of the end points, averaging the end points, linearly connect the ends, or by using some other function to connect the end points, possibly weighting the equation based on qualities at the points.

Data here are based on the earliest end point is extended to fill in the missing data. This makes the most sense since the later end point is not known until connection is restored to the ticket seller. The resulting data can be seen in Tables 15 and 16.

<table>
<thead>
<tr>
<th></th>
<th>BOS-SFO</th>
<th>CVG-BZN</th>
<th>CVG-SFO</th>
<th>DAY-SFO</th>
<th>LAX-BOS</th>
<th>SEA-IAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delta</td>
<td>$-15.63</td>
<td>$-1.22</td>
<td>$61.37</td>
<td>$-50.85</td>
<td>$40.67</td>
<td>$21.60</td>
</tr>
<tr>
<td>Expedia</td>
<td>$40.49</td>
<td>$51.27</td>
<td>$104.41</td>
<td>$57.58</td>
<td>$40.90</td>
<td>$-44.18</td>
</tr>
</tbody>
</table>

Table 15: Ticket cost of algorithm ticket price minus the initial ticket price average after data replacement

<table>
<thead>
<tr>
<th></th>
<th>Minimum Price</th>
<th>Initial Price</th>
<th>Secretary Price</th>
<th>Algorithm Price</th>
<th>Secretary over Initial</th>
<th>Algorithm over Initial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>$485.03</td>
<td>$577.05</td>
<td>$817.13</td>
<td>$581.23</td>
<td>$240.08</td>
<td>$4.18</td>
</tr>
<tr>
<td>Adjusted</td>
<td>$475.25</td>
<td>$569.61</td>
<td>$802.09</td>
<td>$572.49</td>
<td>$232.48</td>
<td>$2.88</td>
</tr>
<tr>
<td>Difference</td>
<td>$9.78</td>
<td>$7.44</td>
<td>$15.04</td>
<td>$8.74</td>
<td>$7.60</td>
<td>$1.30</td>
</tr>
</tbody>
</table>

Table 16: Average ticket purchase prices per customer per route and subsequent calculations
The algorithm ticket price over the initial ticket price is an average loss of $2.88. This is a little smaller loss in the cumulative savings of $4.18 but with a different distribution of the customers whom saved as seen shortly. It is noted that all of the averages did decrease.

The addition of data points will most likely shift the boundary, but the direction the boundary changes depends on the proportion of the missing data. Another interesting effect is the considerations on either side of the boundary: before the boundary the min could be higher or lower and the movement of the boundary could absorb or reveal new plateaus after the new location of the boundary. In figure 20 the algorithm determined purchase price increases in two places because the increased number of prices moves the boundary earlier in time. In figure 21 the algorithm determined a decreased purchase price early on because of an increase in the number of tickets considered. In figure 22 the price decreased because the boundary passed the plateau.
Figure 20: Original algorithm results (light blue) and adjusted algorithm results (yellow) example

Figure 21: Original algorithm results (light blue) and adjusted algorithm results (yellow) example
In summary, this section illustration for the corrections done for imperfect data collection and the overlap between Expedia and the other airlines considered in this study, Continental and Delta. This section also showed how lead time affected the price on ticket purchases; there was not one lead time that dominated, but there were two dominate categories: one to give weeks and greater than three weeks were good times to begin looking for an airline ticket. It was also observed that ticket prices do indeed follow a bucket nature: there are not many different prices offered for a given airline and route.
4.2.3 Data Trend Results

The trends of the data are examined. There have been papers published looking at the trends of airline data [Piga, 2006; McAfee, 2006]. McAfee [2006] studied the following:

1. Prices fall as takeoff approaches,
2. Prices are rising initially,
3. Competition reduces variance of prices,
4. Prices change with each seat sold [McAfee, 2006].

Piga [2006] looks at the following trends:

1. Do fares increase monotonically as the date of departure approaches?
2. Do fares change frequently? [Piga, 2006].

The following study encompasses many of the previously studied questions and provides additional insight. Specifically, there are four types of trends studied here: the occurrences of positive spikes, positive jumps, negative spikes, and negative jumps. A positive spike is defined as an increase in price that does not stay elevated for more than four time periods. A positive jump is defined as an increase in price that stays increased for more than four time periods. Negative spikes and jumps are in the opposite direction.

The following patterns were observed in the trends:

1. There is more activity in middle than beginning and end (see figure 23) (hypothesis II).
2. There is increasing activity as the departure date approaches (see figure 24) (hypothesis II, IVii).
3. There is decreasing activity as the departure date approaches (see figure 25) (hypothesis II, IVii).

4. The seven, fourteen, twenty-one, twenty-eight, and thirty-five day boundaries are observed (see figure 26) (hypothesis II).

5. There are changes under the seven day boundary (see figure 27) (hypothesis III).

Each figure shows the cost difference between the previous point and the point at the labeled point for each date having the characteristic illustrated by the graph. These patterns are observed in varying frequencies. Some data show more than one of the above patterns. For example, one graph may show increased activity in the middle of the data and increased activity under the seven-day boundary (figure 23). The most common is an increase in activity as the departure data approaches. The pattern of boundaries is confirmed in some of the data, but not consistently. Not all of the boundaries are observed in the data; one to three of the boundaries are observed at a time. As discussed earlier, this is probably because the sales were not as good as expected on some of those boundary days and the sales were good on other days, so a large jump in price is not necessary. See appendix 7.6 for graphs representing tables of this data.
Figure 23: Trend example with more activity in the middle

Figure 24: Trend example with an increase in activity as the departure date approaches
Figure 25: Decrease in activity as the departure date approaches

Figure 26: Trend example showing increased activity at boundaries: one week, three week, and four week
Finally, a graph is shown (figure 27) with multiples of seven day boundaries highlighted (hypothesis IViii). One can see that there are changes in the data at each of those boundaries in this example. See appendix 7.4 for the rest of the seven-day boundary graphs.

Figure 27: Seven-day boundary graph for every website over the route BOS to SFO
Figure 28: Trend example with more activity under the seven-day boundary

The above graphs cover the relevant hypotheses as illustrated.
5 Conclusion and Future Work

This chapter contains the conclusions and future work.

5.1 Conclusion

Ticket pricing data was successfully gathered about six routes: BOS-SFO, CVG-BZN, CVG-SFO, DAY-SFO, LAX-BOS, and SEA-IAD and from three websites: Delta, Continental, and Expedia. As illustrated, using an adapted solution from the Secretary Problem saved roughly half of the customers an average of $37.62. The remaining customers lose an average of $54.44 using this method. The cumulative total average is a loss of $4.18 per customer. The lead time study illustrates that it varies by flight whether a long lead time or a short lead time is beneficial. In some cases, one to three weeks is beneficial and in other cases four to five weeks and more is beneficial. One to three weeks is beneficial when there is a price drop, otherwise a longer period is advised. The data trends study gives a result that one might expect: it is most common that there is an increased amount of activity as the departure date approaches. This means that the most money can be saved and the most money can be lost as the departure data approaches. The trend study also shows increased activity around the boundary dates, though it is a gamble that the price will increase or it will decrease. It is illustrated that the data does follow the bucket pattern (hypothesis VI). There were generally three to six buckets. One word of caution to the airfare shopper is that there are outliers – sharp spikes in the price – that consumers should be aware of and avoid when shopping for airline tickets.
5.2 Research Questions Conclusion

This section contains the responses to the research hypothesis stated at the beginning of this work.

5.2.1 Secretary Problem Study

I. Hypothesis

The solution to the Secretary Problem can save consumers money.

Response

The solution to the Secretary Problem, an optimal stopping technique, was slightly modified to include heuristics and found to save approximately half of the customers an average of $37.62. Many consumers benefit from this technique because it leverages the steep drop in price seen in the data. One natural extension of this work is to study many airline flight numbers on many different days to be able to predict when the price drops occur. An algorithm like this one can use those drops in price to its advantage; in the other cases propose purchasing an airline ticket “early enough” when there is not a drop in price predicted. Another extension to this work is to come up with either heuristics or an addition to the algorithm when there is not a drop in price predicted for a given flight to determine how early to purchase a ticket. It is suspected, in the case of a flight without a steep drop, purchasing a ticket early is the best strategy, but more research should be done in this area. There are occasions where the monotonically increasing price is broken by a small, brief decrease in price.
5.2.2 Trends Study

II. **Hypothesis**

The prices of airline tickets monotonically increase as time progresses towards the departure date.

**Response**

The prices of airline tickets do not change in a monotonically increasing fashion as the departure date approaches: about half of the time there is a drop in price approximately one and a half to two weeks before departure and three and a half to four weeks before departure. Thus, two research questions arise: how does one predict when the non-monotonically increasing cases will occur during a given flight and how does one predict the time during a given flight offering at which the decrease will occur. The first point is future work and the second point is looked at by the other research question in this thesis.

III. **Hypothesis**

The final week before departure is the worst week to buy tickets.

**Response**

Generally, after the drop in price at either three and a half to four weeks out or one and a half to two weeks out the prices resume increasing monotonically. In the data collected, there are rare exceptions where a small amount of money could be saved in the final week if a consumer only began looking for a ticket in the final week. These exceptions
could be investigated for a trend.

IV. **Hypothesis**

The following related questions are on the topic of price volatility.

i) The ticket prices change frequently. This question is confirming the literature such as McCartney’s [1997] article which discusses airlines’ use of technology to get as much money as possible from customers. The article illustrates the change in ticket prices as the departure date approaches and tickets are purchased.

ii) The price changes more often as the departure date approaches. This is to confirm the statement by Piga [2006] that fares are less stable when closer to the departure date.

iii) At the end of each seven-day interval from departure there is a sudden change in price. This question comes from the literature and speculation around the 28-day, 21-day, 14-day, and 7-day boundaries [Piga, 2006; SoYouWanna.com, 2006].

**Response**

i) The ticket prices do indeed change frequently. This is evident from looking at the data, as seen in figure 11, as well as the subsequent analysis on the trends of the data in figures 19 through 23. In the data captured, there was a cumulative average of 2.33 changes in price per day, with a range of 0 to 23 changes in a given day.
ii) There is often an increased volatility as the departure date gets closer, as seen in figure 25. There is also an additional case seen in the data: in figure 24, it is evident that there is more activity in the middle of the time period than earlier or later in the time period. This might be explained by most consumers purchasing their tickets in the middle of the time period, and fewer purchasing earlier or later. The conclusion is that consumers should be wary of the price being offered in the middle of the time period.

iii) At the end of each seven-day interval from departure there is a sudden change in price as evident by figure 23.

In summary: with this knowledge, the consumer should use a strategy that involves avoiding the hikes in ticket price. There is increased volatility as the departure date gets closer and at the end of each seven-day interval.

V. Hypothesis

A larger lead time allows for greater airline ticket savings.

Response

Lead time is important for the consumer for two reasons: in the cases where there is a drop in price, the consumer must begin looking early enough to catch the drop in price; otherwise, when there is not a drop in price, the consumer should begin shopping early to catch a cheaper price. Further research is needed to clearly identify the cases where a drop will occur and one will not occur.
VI. **Hypothesis**

There are a small number of unique prices for a given flight. This is following statements in the literature such as the one given by McCartney [1997] concerning the tendency for there to be few given ticket prices.

**Response**

There are indeed a small number of unique prices for a given flight if a band of +/- 5% is put on a price range. There are typically three to six tight ranges of prices per flight. This information could be used to simplify the construction of an algorithm to differentiate prices.

### 5.3 Contribution To Knowledge

The key contribution of this work is the recognition of a simple way for consumers to heuristically time the purchase of a cheaper airline ticket. When starting to look for an airline ticket three to four weeks out or so, 37% of the remaining time (a little more than a third of the remaining time) before departure is used to gather knowledge of the offered prices. After that period has expired, the next airline ticket offered that is cheaper than all of those seen so far is a good choice. Alternatively, it could be an airline that does not have large down swings, so it is recommended to purchase a ticket a few hours after the price rises significantly and does not return to near the original price, since it is not expected to return later. This technique is simple for a consumer to implement since there are only a few steps. Other methods, such as those from machine learning, are
considerably more complex and users may not readily trust them because of their complexity and tendency to change in unexpected ways.

The second major contribution this work provides is the gathering and analysis of airline ticket pricing data. There are very few articles that have looked at airline ticket data. Most of them look at the final purchase prices over time instead of the offered prices for each departure over time. This thesis looked at the following: lead time, data buckets, changes per day, and the overlap between an online travel agent and two airlines. Six routes were considered. These give some insight into how the airlines price their tickets and what might be used to look for the cheaper tickets.

Finally, an overview of the airline industry and an overview of different possible technical approaches to this problem are given.

5.4 Future Work

This section provides thoughts on possible future work related to this exploratory study. With more data, more can be learned. The major missing component to understanding airline data trends is an insight into the demand, which can be inferred from the number of seats left on a given flight over time. Since this information is not available, a lot of data must be gathered and examined to work around this missing variable. A large standardized data set should be gathered for many different flight numbers on different days, compensating for seasonal and holiday demand changes.
One trend noticed in the data is a sudden drop in price in about half of the routes plotted. A study to uncover the routes with this sudden drop in price would give customers valuable insight into money savings. A related study is to predict the time before departure the drop will occur. These studies could be done by collecting a vast amount of routes for a given flight number and looking for a pattern in the drops. When it is predicted that a sudden drop will not occur the customer could be alerted to purchase a ticket 28-days or more out from departure.

Additionally, the solution to the adapted Secretary Problem used here could be further improved. The restriction that the observed price after the boundary be less than all previous prices could be relaxed to be some other value, such as lower than the average price before the boundary. One problem recognized is that a short-lived decrease in price before the boundary has a very negative influence on the rest of the algorithm in that it has to find a price less than that negative spike.

Further, the trends of the data could be studied in more detail. For the four trends studied, for example, there are unexplained spikes, and it would be interesting to determine if there is a pattern to those unexplained spikes. It would also be interesting to further examine four trends studied by looking for additional patterns.
5.5 Final Remarks

This exploratory study helped gain an insight into purchasing an airline ticket. An exploratory study was useful in showing the complexity of the airline ticket pricing gathered; it can change rapidly without warning for a given flight. The study also showed how money could potentially be saved. Another study to look over past months and years for a pattern for the sudden, large drops in price could be very interesting. Money was saved here over buying the first ticket seen by a given customer when there was a drop in price; prediction of these large drops could determine whether to purchase early or wait for the large savings.
6 References

6.1 Technical References


Bernacki, M. and P. Wlodarczyk, [2005], “Neural networks and artificial intelligence in electronics,” 2005, Virtual Laboratory of Artificial Intelligence, University of Science and Technology, Krakow, Poland


Lewis, R. J., [2000], "An Introduction to Classification and Regression Tree (CART) Analysis," May, 2000, *Annual Meeting of the Society for Academic Emergency Medicine Presentation*


McAfee, R. P. and V. te Velde, [2006], “Dynamic Pricing in the Airline Industry,”
December, 2006 Handbook on Economics and Information Systems, Ed: T.J. Hendershott,
Elsevier, forthcoming


Moving Average, [2007], February 27, 2007, In Wikipedia, The Free Encyclopedia,


Pedersen, J. L., [2007], “An Optimal Selling Strategy for Stock Trading Based on Predicting the Maximum Price,” August, 2007, Laboratory of Actuarial Mathematics, Department of Mathematical Sciences, University of Copenhagen

Postel, J. B., [1982], “Simple Mail Transfer Protocol,” August, 1982, RFC 821


Quinlan, J. R., [1990], “Learning Logical Definitions from Relations,” August, 1990, Machine Learning, volume 5, issue 3, p. 239 – 266


Sunzu, J., [2007], “American Options and Optimal Stopping Problems,” June, 2007, African Institute for Mathematical Sciences (AIMS), University of Witwatersrand


Ting, K. M. and I. H. Witten, [1999], “Issues in stacked generalization,” May 1999, 
*Journal of Artificial Intelligence Research*, volume 10, p. 271 – 289


Ulrich, J., [2006], “Online Combinatorial Auctions,” December, 2006, Department of Computer Science, University of British Columbia


6.2 Trade and Popular Press References


Ashley, M., [2006], “Mythbustin’: Is Wednesday at midnight the best time to buy airline tickets,” August 9, 2006, UpgradeTravel, retrieved December 18, 2006 from http://upgradetravel.blogspot.com/2006/08/mythbustin-is-wednesday-at-midnight.html


Fehrenbacher, K., [2006], “TripHub: Your Mom’s Trip Planner,” August 8, 2006, 

November 2, 2006 from http://www.artoftravel.com/04cheapflights.htm

Kirkpatrick, M., [2006], “Farecast Selling Airline Ticket Price Guarantees,” November 13, 
2006, Tech Crunch, retrieved December 11, 2006 from  

Livingston, B., [2006], “Farecast Now Predicts Prices for 55 Cities,” August 29, 2006, 
Datamation, Earthweb, retrieved December 21, 2006 from  

Marshall, M., [2006], “Farecast expands airfare predict; just know when to use them,” 
September 24, 2006, Venture Beat, retrieved December 1, 2006 from  
http://venturebeat.com/2006/09/24/farecast-expands-airfare-predictions-just-know-when-to-use-them/

McCartney, S., [1997], “Airlines Rely on Technology To Manipulate Fare Structure,” 
November 3, 1997, Wall Street Journal


Seaney, R., [2006], “When is the Best Time to Buy Air Travel?” August 14, 2006, FareCompare, retrieved November 28, 2006, from http://www.farecompare.com/downloads/WhenIsTheBestTimeToBuyAirTravel/WhenIsTheBestTimeToBuyAirTravel.htm?home=yes


7 Appendices

7.1 Combined Route Graphs

This appendix section contains graphs for the combined gathered data for each route including Secretary Algorithm results from the point of view of the first customer.
DAY-SFO Combined

- **X** - Best before boundary
- **X** - Cheapest
- **X** - Secretary result

<table>
<thead>
<tr>
<th>Date</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wed Apr 12</td>
<td>0</td>
</tr>
<tr>
<td>Thu Apr 13</td>
<td>0</td>
</tr>
<tr>
<td>Fri Apr 14</td>
<td>0</td>
</tr>
<tr>
<td>Sat Apr 15</td>
<td>0</td>
</tr>
<tr>
<td>Sun Apr 16</td>
<td>0</td>
</tr>
<tr>
<td>Mon Apr 17</td>
<td>0</td>
</tr>
<tr>
<td>Tue Apr 18</td>
<td>0</td>
</tr>
<tr>
<td>Wed Apr 19</td>
<td>0</td>
</tr>
<tr>
<td>Thu Apr 20</td>
<td>0</td>
</tr>
<tr>
<td>Fri Apr 21</td>
<td>0</td>
</tr>
<tr>
<td>Sat Apr 22</td>
<td>0</td>
</tr>
<tr>
<td>Sun Apr 23</td>
<td>0</td>
</tr>
<tr>
<td>Mon Apr 24</td>
<td>0</td>
</tr>
<tr>
<td>Tue Apr 25</td>
<td>0</td>
</tr>
<tr>
<td>Wed Apr 26</td>
<td>0</td>
</tr>
<tr>
<td>Thu Apr 27</td>
<td>0</td>
</tr>
<tr>
<td>Fri Apr 28</td>
<td>0</td>
</tr>
<tr>
<td>Sat Apr 29</td>
<td>0</td>
</tr>
<tr>
<td>Sun Apr 30</td>
<td>0</td>
</tr>
<tr>
<td>Mon May 01</td>
<td>0</td>
</tr>
<tr>
<td>Tue May 02</td>
<td>0</td>
</tr>
<tr>
<td>Wed May 03</td>
<td>0</td>
</tr>
<tr>
<td>Thu May 04</td>
<td>0</td>
</tr>
<tr>
<td>Fri May 05</td>
<td>0</td>
</tr>
<tr>
<td>Sat May 06</td>
<td>0</td>
</tr>
<tr>
<td>Sun May 07</td>
<td>0</td>
</tr>
<tr>
<td>Mon May 08</td>
<td>0</td>
</tr>
<tr>
<td>Tue May 09</td>
<td>0</td>
</tr>
<tr>
<td>Wed May 10</td>
<td>0</td>
</tr>
<tr>
<td>Thu May 11</td>
<td>0</td>
</tr>
<tr>
<td>Fri May 12</td>
<td>0</td>
</tr>
<tr>
<td>Sat May 13</td>
<td>0</td>
</tr>
<tr>
<td>Sun May 14</td>
<td>0</td>
</tr>
<tr>
<td>Mon May 15</td>
<td>0</td>
</tr>
<tr>
<td>Tue May 16</td>
<td>0</td>
</tr>
<tr>
<td>Wed May 17</td>
<td>0</td>
</tr>
<tr>
<td>Thu May 18</td>
<td>0</td>
</tr>
<tr>
<td>Fri May 19</td>
<td>0</td>
</tr>
<tr>
<td>Sat May 20</td>
<td>0</td>
</tr>
<tr>
<td>Sun May 21</td>
<td>0</td>
</tr>
<tr>
<td>Mon May 22</td>
<td>0</td>
</tr>
</tbody>
</table>
7.2 Graphs of Secretary Algorithm with a normal distribution of customers

This appendix section contains graphs of each set of data and the Secretary Algorithm results as if there was a customer at each data point – the pink line is the cost of the ticket of the Secretary Algorithm result at that point. Note: Data points are not displayed for the customers that the Secretary Algorithm does not save customers money over the initial ticket price.
Secretary - BOS-SFO Continental

Date

Original • Secretary
Secretary - CVG-BZN Continental

Date

Price

Original

Secretary
<table>
<thead>
<tr>
<th>Date</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mon Apr 16</td>
<td>300</td>
</tr>
<tr>
<td>Mon Apr 17</td>
<td>350</td>
</tr>
<tr>
<td>Mon Apr 18</td>
<td>400</td>
</tr>
<tr>
<td>Mon Apr 19</td>
<td>450</td>
</tr>
<tr>
<td>Mon Apr 20</td>
<td>500</td>
</tr>
<tr>
<td>Mon Apr 21</td>
<td>550</td>
</tr>
<tr>
<td>Mon Apr 22</td>
<td>600</td>
</tr>
<tr>
<td>Mon Apr 23</td>
<td>650</td>
</tr>
<tr>
<td>Mon Apr 24</td>
<td>700</td>
</tr>
<tr>
<td>Mon Apr 25</td>
<td>750</td>
</tr>
<tr>
<td>Mon Apr 26</td>
<td>800</td>
</tr>
<tr>
<td>Mon Apr 27</td>
<td>850</td>
</tr>
<tr>
<td>Mon Apr 28</td>
<td>900</td>
</tr>
<tr>
<td>Mon Apr 29</td>
<td></td>
</tr>
<tr>
<td>Mon Apr 30</td>
<td></td>
</tr>
<tr>
<td>Tue May 01</td>
<td></td>
</tr>
<tr>
<td>Tue May 02</td>
<td></td>
</tr>
<tr>
<td>Tue May 03</td>
<td></td>
</tr>
<tr>
<td>Tue May 04</td>
<td></td>
</tr>
<tr>
<td>Tue May 05</td>
<td></td>
</tr>
<tr>
<td>Tue May 06</td>
<td></td>
</tr>
<tr>
<td>Tue May 07</td>
<td></td>
</tr>
<tr>
<td>Tue May 08</td>
<td></td>
</tr>
<tr>
<td>Tue May 09</td>
<td></td>
</tr>
<tr>
<td>Tue May 10</td>
<td></td>
</tr>
<tr>
<td>Tue May 11</td>
<td></td>
</tr>
<tr>
<td>Tue May 12</td>
<td></td>
</tr>
<tr>
<td>Tue May 13</td>
<td></td>
</tr>
<tr>
<td>Tue May 14</td>
<td></td>
</tr>
<tr>
<td>Tue May 15</td>
<td></td>
</tr>
<tr>
<td>Tue May 16</td>
<td></td>
</tr>
<tr>
<td>Tue May 17</td>
<td></td>
</tr>
<tr>
<td>Tue May 18</td>
<td></td>
</tr>
<tr>
<td>Tue May 19</td>
<td></td>
</tr>
<tr>
<td>Tue May 20</td>
<td></td>
</tr>
<tr>
<td>Tue May 21</td>
<td></td>
</tr>
<tr>
<td>Tue May 22</td>
<td></td>
</tr>
<tr>
<td>Tue May 23</td>
<td></td>
</tr>
<tr>
<td>Tue May 24</td>
<td></td>
</tr>
<tr>
<td>Tue May 25</td>
<td></td>
</tr>
<tr>
<td>Tue May 26</td>
<td></td>
</tr>
<tr>
<td>Tue May 27</td>
<td></td>
</tr>
<tr>
<td>Tue May 28</td>
<td></td>
</tr>
<tr>
<td>Tue May 29</td>
<td></td>
</tr>
<tr>
<td>Tue May 30</td>
<td></td>
</tr>
<tr>
<td>Wed May 31</td>
<td></td>
</tr>
</tbody>
</table>
Secretary - DAY-SFO Continental

<table>
<thead>
<tr>
<th>Date</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wed Apr 12</td>
<td>09:00</td>
</tr>
<tr>
<td>Thu Apr 13</td>
<td>04:00</td>
</tr>
<tr>
<td>Fri Apr 14</td>
<td>18:00</td>
</tr>
<tr>
<td>Sat Apr 15</td>
<td>13:00</td>
</tr>
<tr>
<td>Sun Apr 16</td>
<td>08:00</td>
</tr>
<tr>
<td>Mon Apr 17</td>
<td>05:00</td>
</tr>
<tr>
<td>Tue Apr 18</td>
<td>02:00</td>
</tr>
<tr>
<td>Wed Apr 19</td>
<td>00:00</td>
</tr>
<tr>
<td>Thu Apr 20</td>
<td>18:00</td>
</tr>
<tr>
<td>Fri Apr 21</td>
<td>19:00</td>
</tr>
<tr>
<td>Sat Apr 22</td>
<td>14:00</td>
</tr>
<tr>
<td>Sun Apr 23</td>
<td>09:00</td>
</tr>
<tr>
<td>Mon Apr 24</td>
<td>06:00</td>
</tr>
<tr>
<td>Tue Apr 25</td>
<td>11:00</td>
</tr>
<tr>
<td>Wed Apr 26</td>
<td>10:00</td>
</tr>
<tr>
<td>Thu Apr 27</td>
<td>13:00</td>
</tr>
<tr>
<td>Fri Apr 28</td>
<td>10:00</td>
</tr>
<tr>
<td>Sat Apr 29</td>
<td>15:00</td>
</tr>
<tr>
<td>Sun Apr 30</td>
<td>15:00</td>
</tr>
<tr>
<td>Mon May  1</td>
<td>20:00</td>
</tr>
<tr>
<td>Tue May  2</td>
<td>19:00</td>
</tr>
<tr>
<td>Wed May  3</td>
<td>18:00</td>
</tr>
<tr>
<td>Thu May  4</td>
<td>20:00</td>
</tr>
<tr>
<td>Fri May  5</td>
<td>19:00</td>
</tr>
<tr>
<td>Sat May  6</td>
<td>20:00</td>
</tr>
<tr>
<td>Sun May  7</td>
<td>15:00</td>
</tr>
<tr>
<td>Mon May  8</td>
<td>12:00</td>
</tr>
<tr>
<td>Tue May  9</td>
<td>12:00</td>
</tr>
<tr>
<td>Wed May 10</td>
<td>07:00</td>
</tr>
<tr>
<td>Thu May 11</td>
<td>02:00</td>
</tr>
<tr>
<td>Fri May 12</td>
<td>01:00</td>
</tr>
<tr>
<td>Sat May 13</td>
<td>06:00</td>
</tr>
<tr>
<td>Sun May 14</td>
<td>08:00</td>
</tr>
<tr>
<td>Mon May 15</td>
<td>11:00</td>
</tr>
<tr>
<td>Tue May 16</td>
<td>10:00</td>
</tr>
<tr>
<td>Wed May 17</td>
<td>09:00</td>
</tr>
<tr>
<td>Thu May 18</td>
<td>08:00</td>
</tr>
<tr>
<td>Fri May 19</td>
<td>07:00</td>
</tr>
<tr>
<td>Sat May 20</td>
<td>06:00</td>
</tr>
<tr>
<td>Sun May 21</td>
<td>05:00</td>
</tr>
</tbody>
</table>

[Graph showing price changes over time]
<table>
<thead>
<tr>
<th>Date</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wed Apr 12</td>
<td>300</td>
</tr>
<tr>
<td>Thu Apr 13</td>
<td>400</td>
</tr>
<tr>
<td>Fri Apr 14</td>
<td>500</td>
</tr>
<tr>
<td>Sat Apr 15</td>
<td>600</td>
</tr>
<tr>
<td>Sun Apr 16</td>
<td>700</td>
</tr>
<tr>
<td>Mon Apr 17</td>
<td>800</td>
</tr>
<tr>
<td>Tue Apr 18</td>
<td>900</td>
</tr>
</tbody>
</table>

**Secretary - BOS-SFO Expedia**

- **Original**
- **Secretary**
7.3 Graphs of an adapted algorithm including the Secretary Algorithm

This section contains a graphical summary of an adapted algorithm along with the Secretary Algorithm results.
Algorithm - DAY-SFO Delta

<table>
<thead>
<tr>
<th>Date</th>
<th>Price</th>
<th>Original</th>
<th>Secretary</th>
<th>Algo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wed Apr 12 09:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thu Apr 13 04:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fri Apr 14 13:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sat Apr 15 13:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sun Apr 16 08:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mon Apr 17 03:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tue Apr 18 05:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wed Apr 19 23:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thu Apr 20 18:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fri Apr 21 19:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sat Apr 22 14:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sun Apr 23 09:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mon Apr 24 04:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tue Apr 25 09:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wed Apr 26 21:10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thu Apr 27 18:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fri Apr 28 15:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sat Apr 29 11:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sun Apr 30 07:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mon May 01 02:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tue May 02 07:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wed May 03 02:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thu May 04 07:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fri May 05 03:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sat May 06 06:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sun May 07 03:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mon May 08 11:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tue May 09 21:10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wed May 10 02:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thu May 11 10:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fri May 12 07:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sat May 13 06:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sun May 14 08:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mon May 15 02:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tue May 16 09:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wed May 17 23:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thu May 18 10:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fri May 19 15:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sat May 20 20:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sun May 21 21:50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
7.4 Seven-day boundary graphs

This section contains graphs with gathered data combined for each route showing the seven, fourteen, twenty-one, and twenty-eight day boundaries.
7.5 Tables for buckets

This section contains the data for the buckets, representing the number of general levels of ticket pricing.

Note: A tier contains prices 5% higher or lower from the recorded price.
### Delta Buckets

**results_Delta_BOS-SFO.txt**

<table>
<thead>
<tr>
<th>Price ($)</th>
<th># of points</th>
</tr>
</thead>
<tbody>
<tr>
<td>619.21</td>
<td>962</td>
</tr>
<tr>
<td>479.2</td>
<td>259</td>
</tr>
<tr>
<td>779.21</td>
<td>97</td>
</tr>
<tr>
<td>694.2</td>
<td>29</td>
</tr>
<tr>
<td>529.2</td>
<td>7</td>
</tr>
<tr>
<td>879.21</td>
<td>1</td>
</tr>
</tbody>
</table>

**results_Delta_CVG-SFO.txt**

<table>
<thead>
<tr>
<th>Price ($)</th>
<th># of points</th>
</tr>
</thead>
<tbody>
<tr>
<td>569.6</td>
<td>718</td>
</tr>
<tr>
<td>679.6</td>
<td>437</td>
</tr>
<tr>
<td>1124.09</td>
<td>87</td>
</tr>
<tr>
<td>1318.6</td>
<td>73</td>
</tr>
<tr>
<td>779.1</td>
<td>21</td>
</tr>
<tr>
<td>974.09</td>
<td>11</td>
</tr>
</tbody>
</table>

**results_Delta_CVG-BZN.txt**

<table>
<thead>
<tr>
<th>Price ($)</th>
<th># of points</th>
</tr>
</thead>
<tbody>
<tr>
<td>743.7</td>
<td>638</td>
</tr>
<tr>
<td>1312.2</td>
<td>481</td>
</tr>
<tr>
<td>663.71</td>
<td>353</td>
</tr>
<tr>
<td>1137.71</td>
<td>260</td>
</tr>
<tr>
<td>935.7</td>
<td>178</td>
</tr>
<tr>
<td>835.5</td>
<td>132</td>
</tr>
<tr>
<td>1030</td>
<td>40</td>
</tr>
<tr>
<td>453.7</td>
<td>26</td>
</tr>
<tr>
<td>1990</td>
<td>3</td>
</tr>
<tr>
<td>513.7</td>
<td>2</td>
</tr>
<tr>
<td>1600</td>
<td>1</td>
</tr>
<tr>
<td>Price ($)</td>
<td># of points</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>844.19</td>
<td>602</td>
</tr>
<tr>
<td>704.2</td>
<td>344</td>
</tr>
<tr>
<td>931.2</td>
<td>226</td>
</tr>
<tr>
<td>1118.21</td>
<td>79</td>
</tr>
<tr>
<td>559.2</td>
<td>59</td>
</tr>
<tr>
<td>504.2</td>
<td>40</td>
</tr>
<tr>
<td>1248.21</td>
<td>2</td>
</tr>
<tr>
<td>1590.5</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Price ($)</th>
<th># of points</th>
</tr>
</thead>
<tbody>
<tr>
<td>358.6</td>
<td>615</td>
</tr>
<tr>
<td>403.6</td>
<td>297</td>
</tr>
<tr>
<td>558.61</td>
<td>198</td>
</tr>
<tr>
<td>718.6</td>
<td>147</td>
</tr>
<tr>
<td>448.6</td>
<td>91</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Price ($)</th>
<th># of points</th>
</tr>
</thead>
<tbody>
<tr>
<td>689.59</td>
<td>966</td>
</tr>
<tr>
<td>817.61</td>
<td>293</td>
</tr>
<tr>
<td>946.61</td>
<td>79</td>
</tr>
</tbody>
</table>
Continental Buckets

<table>
<thead>
<tr>
<th>Price ($)</th>
<th># of points</th>
</tr>
</thead>
<tbody>
<tr>
<td>540.2</td>
<td>784</td>
</tr>
<tr>
<td>484.2</td>
<td>226</td>
</tr>
<tr>
<td>634.2</td>
<td>151</td>
</tr>
<tr>
<td>399.8</td>
<td>84</td>
</tr>
<tr>
<td>765.5</td>
<td>75</td>
</tr>
<tr>
<td>359.8</td>
<td>28</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Price ($)</th>
<th># of points</th>
</tr>
</thead>
<tbody>
<tr>
<td>540.2</td>
<td>784</td>
</tr>
<tr>
<td>484.2</td>
<td>226</td>
</tr>
<tr>
<td>634.2</td>
<td>151</td>
</tr>
<tr>
<td>399.8</td>
<td>84</td>
</tr>
<tr>
<td>765.5</td>
<td>75</td>
</tr>
<tr>
<td>359.8</td>
<td>27</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Price ($)</th>
<th># of points</th>
</tr>
</thead>
<tbody>
<tr>
<td>458.7</td>
<td>601</td>
</tr>
<tr>
<td>536.2</td>
<td>487</td>
</tr>
<tr>
<td>671.2</td>
<td>418</td>
</tr>
<tr>
<td>597</td>
<td>417</td>
</tr>
<tr>
<td>389.8</td>
<td>92</td>
</tr>
<tr>
<td>757.7</td>
<td>61</td>
</tr>
<tr>
<td>850.5</td>
<td>18</td>
</tr>
<tr>
<td>Price ($)</td>
<td># of points</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>540.2</td>
<td>790</td>
</tr>
<tr>
<td>484.2</td>
<td>227</td>
</tr>
<tr>
<td>634.2</td>
<td>153</td>
</tr>
<tr>
<td>399.8</td>
<td>90</td>
</tr>
<tr>
<td>765.5</td>
<td>78</td>
</tr>
<tr>
<td>359.8</td>
<td>27</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Price ($)</th>
<th># of points</th>
</tr>
</thead>
<tbody>
<tr>
<td>540.2</td>
<td>783</td>
</tr>
<tr>
<td>484.2</td>
<td>228</td>
</tr>
<tr>
<td>634.2</td>
<td>151</td>
</tr>
<tr>
<td>399.8</td>
<td>83</td>
</tr>
<tr>
<td>765.5</td>
<td>75</td>
</tr>
<tr>
<td>359.8</td>
<td>28</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Price ($)</th>
<th># of points</th>
</tr>
</thead>
<tbody>
<tr>
<td>540.2</td>
<td>778</td>
</tr>
<tr>
<td>484.2</td>
<td>228</td>
</tr>
<tr>
<td>634.2</td>
<td>151</td>
</tr>
<tr>
<td>399.8</td>
<td>86</td>
</tr>
<tr>
<td>765.5</td>
<td>76</td>
</tr>
<tr>
<td>359.8</td>
<td>27</td>
</tr>
</tbody>
</table>
### Expedia Buckets

#### results_Expedia_BOS-SFO.txt

<table>
<thead>
<tr>
<th>Price ($)</th>
<th># of points</th>
</tr>
</thead>
<tbody>
<tr>
<td>370.2</td>
<td>728</td>
</tr>
<tr>
<td>448.89</td>
<td>353</td>
</tr>
<tr>
<td>527.21</td>
<td>198</td>
</tr>
<tr>
<td>617.21</td>
<td>76</td>
</tr>
<tr>
<td>858.9</td>
<td>1</td>
</tr>
</tbody>
</table>

#### results_Expedia_CVG-SFO.txt

<table>
<thead>
<tr>
<th>Price ($)</th>
<th># of points</th>
</tr>
</thead>
<tbody>
<tr>
<td>544.9</td>
<td>407</td>
</tr>
<tr>
<td>313.7</td>
<td>326</td>
</tr>
<tr>
<td>388.19</td>
<td>227</td>
</tr>
<tr>
<td>602.2</td>
<td>129</td>
</tr>
<tr>
<td>925.39</td>
<td>84</td>
</tr>
<tr>
<td>764.11</td>
<td>76</td>
</tr>
<tr>
<td>663.7</td>
<td>50</td>
</tr>
<tr>
<td>450.2</td>
<td>44</td>
</tr>
<tr>
<td>1153.11</td>
<td>1</td>
</tr>
</tbody>
</table>

#### results_Expedia_CVG-BZN.txt

<table>
<thead>
<tr>
<th>Price ($)</th>
<th># of points</th>
</tr>
</thead>
<tbody>
<tr>
<td>475.7</td>
<td>759</td>
</tr>
<tr>
<td>1242.7</td>
<td>434</td>
</tr>
<tr>
<td>410.6</td>
<td>202</td>
</tr>
<tr>
<td>943.2</td>
<td>189</td>
</tr>
<tr>
<td>1042.1</td>
<td>171</td>
</tr>
<tr>
<td>715.3</td>
<td>169</td>
</tr>
<tr>
<td>593.7</td>
<td>52</td>
</tr>
<tr>
<td>793.2</td>
<td>46</td>
</tr>
<tr>
<td>526.8</td>
<td>16</td>
</tr>
<tr>
<td>362.6</td>
<td>1</td>
</tr>
<tr>
<td>1574.3</td>
<td>1</td>
</tr>
<tr>
<td>Price ($)</td>
<td># of points</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>318.2</td>
<td>627</td>
</tr>
<tr>
<td>384.2</td>
<td>257</td>
</tr>
<tr>
<td>592.7</td>
<td>218</td>
</tr>
<tr>
<td>266.2</td>
<td>113</td>
</tr>
<tr>
<td>431.5</td>
<td>85</td>
</tr>
<tr>
<td>490.5</td>
<td>38</td>
</tr>
<tr>
<td>794.79</td>
<td>21</td>
</tr>
<tr>
<td>671.5</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Price ($)</th>
<th># of points</th>
</tr>
</thead>
<tbody>
<tr>
<td>482.6</td>
<td>752</td>
</tr>
<tr>
<td>569.2</td>
<td>204</td>
</tr>
<tr>
<td>403.9</td>
<td>150</td>
</tr>
<tr>
<td>690.5</td>
<td>87</td>
</tr>
<tr>
<td>984.8</td>
<td>62</td>
</tr>
<tr>
<td>286.2</td>
<td>49</td>
</tr>
<tr>
<td>346.2</td>
<td>32</td>
</tr>
<tr>
<td>791</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Price ($)</th>
<th># of points</th>
</tr>
</thead>
<tbody>
<tr>
<td>438.59</td>
<td>334</td>
</tr>
<tr>
<td>527.2</td>
<td>139</td>
</tr>
<tr>
<td>693.9</td>
<td>61</td>
</tr>
<tr>
<td>622.2</td>
<td>1</td>
</tr>
</tbody>
</table>
7.6 Data Trends

This section shows graphs of the positive spikes, positive jumps, negative spikes, and negative jumps of the data. A spike is an increase/decrease in price that lasts four or less data periods, where a period is half an hour. A jump is an increase/decrease that lasts for more than four data periods.
7.6.1 Positive spikes

This subsection shows the positive spikes in the data. A positive spike is an increase in the ticket price that decreases in four time periods or less.
7.6.2 Positive Jumps

This subsection shows the positive jumps in the data. A positive jump is an increase in the ticket cost that does not decrease in four time periods.
Positive Jump - Expedia CVG-BZN

Date

Thu Apr 13 05:30
Sat Apr 15 03:30
Fri Apr 21 03:00
Mon Apr 24 05:00
Mon May 01 05:30
Mon May 08 19:00
Tue May 09 07:00
Tue May 09 15:30
Thu May 11 18:30
Thu May 11 22:00
Fri May 12 17:00
Sat May 13 01:00
Sat May 13 03:00
Thu May 18 13:00
Sat May 20 02:00
Tue May 23 09:30
Wed May 24 01:30

Difference

0
50
100
150
200
250
Positive Jump - Expedia SEA-IAD

Date
 Thu Apr 13 05:30
 Thu Apr 27 09:00
 Thu Apr 27 11:00
 Tue May 02 11:00
 Tue May 02 15:00
 Thu May 04 19:00
 Mon May 08 19:00
 Tue May 09 02:00
 Tue May 09 07:30
 Tue May 09 13:30
 Wed May 10 10:00
 Thu May 11 00:00
 Thu May 11 15:30
 Sat May 13 07:00
 Tue May 16 00:00
 Tue May 16 10:00
 Tue May 16 13:00
 Wed May 17 07:00
 Thu May 17 13:00
 Sat May 19 07:00
 Sat May 19 13:00

Difference
 0
 25
 50
 75
 100
 125
 150
 175
 200
 225
 250

Date
7.6.3 Negative Spikes

This subsection shows the negative spikes in the data. A negative spike is a decrease in the ticket price that increases in four time periods or less.
Negative Spike - Delta LAX-BOS
None
Negative Spike - Continental BOS-SFO
Negative Spike - Continental DAY-SFO

Date

Fri Apr 14 20:00
Thu Apr 20 22:30
Fri Apr 21 21:00
Wed May 03 23:00
Thu May 11 13:00
Wed May 17 14:30
Sat May 20 01:00

Difference
Negative Spike - Expedia LAX-BOS
None
7.6.4 Negative Jumps

This subsection shows the negative jumps. A negative jump is a decrease in the ticket cost that does not increase in four time periods.
Negative Jump - Delta DAY-SFO

Date

Difference
Negative Jump - Expedia SEA-IAD

Date

$300.00
$250.00
$200.00
$150.00
$100.00
$50.00
$0.00

Difference

Wed Apr 26 11:30
Thu Apr 27 08:00
Sat Apr 29 18:00
Sun Apr 30 16:00
Mon May 01 09:30
Mon May 01 14:00
Tue May 02 12:00
Wed May 03 04:30
Wed May 03 13:30
Wed May 03 22:00
Thu May 04 13:00
Thu May 04 18:00
Fri May 05 11:00
Sat May 06 07:30
Sat May 06 10:30
Sun May 07 19:30
Mon May 08 06:00
Mon May 08 07:00
Mon May 08 22:00
Wed May 09 03:00
Wed May 09 04:30
Tue May 10 11:00
Wed May 10 12:00
Wed May 11 16:00
Thu May 11 14:00
Thu May 11 15:00
Mon May 15 18:00
Mon May 15 23:30
Sun May 17 04:30
Sat May 17 15:00
Sun Apr 29 18:00
Sat Apr 29 18:00
Thu Apr 27 08:00
Wed Apr 26 11:30

Date

$300.00
$250.00
$200.00
$150.00
$100.00
$50.00
$0.00

Difference
7.7 Lead Time Study

This section contains graphs for the use in the lead time study. These graphs show the difference between the adapted algorithm ticket price and the initial price a customer could have purchased a ticket.
Algorithm Over Initial - Delta CVG-SFO
Algorithm Over Initial - Delta SEA-IAD
Algorithm Over Initial - Expedia DAY-SFO
Algorithm Over Initial - Expedia SEA-IAD
7.8 Data Extended

For these graphs the missing data in the above graphs have been filled in by extending the previous read data. These graphs also show the original algorithm results and the new extended results.
Algo Comparison - SEA-IAD Delta

<table>
<thead>
<tr>
<th>Date</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td></td>
</tr>
<tr>
<td>Algo</td>
<td></td>
</tr>
<tr>
<td>AlgoOld</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>Price</td>
</tr>
<tr>
<td>--------------</td>
<td>-------</td>
</tr>
<tr>
<td>Original</td>
<td></td>
</tr>
<tr>
<td>Algo</td>
<td></td>
</tr>
<tr>
<td>AlgoOld</td>
<td></td>
</tr>
</tbody>
</table>