Solving Attractiveness-based Stochastic Fleeting by MapReduce

Mingyang Di

Department of Industrial Engineering and Management Sciences
Northwestern University
mingyangdi2012@u.northwestern.edu

Diego Klabjan

Department of Industrial Engineering and Management Sciences
Northwestern University
d-klabjan@northwestern.edu

Sergey Shebalov

Sabre Airline Solutions, Sabre Holdings
sergey.shebalov@sabre-holdings.com

April 5, 2016

Abstract

Fleeting, the assignment of aircraft types, each having a different capacity, to the scheduled flights is an essential component of an airline's overall scheduling process and exerts a huge impact on its revenue. In order to address the high level of uncertainty in the market demand when the fleeting decision is made and to capture the network effects (i.e., spill and recapture) for a more accurate estimate of passenger flow, we present a two-stage stochastic model which incorporates an attractiveness-based spill and recapture framework. Such a model considers spill and recapture based on passenger utility from itineraries. Solution approaches based on a distributed framework, namely MapReduce, are developed to reduce the computational time, and numerical results are reported using real data from a medium-size airline to evaluate and compare the proposed procedures.

Keywords: airline fleeting; attractiveness-based spill and recapture; stochastic programming; parallel computing; MapReduce.

1 Introduction

Airlines around the world continue to face increasing capital and operational costs. As competition intensifies, they have been forced to cut costs and uphold revenue by utilizing their equipment capacity more efficiently to accommodate passenger demand. This is generally known as the *fleet assignment problem*, which deals with assigning aircraft of different capacities to the scheduled flight legs based on availabilities, costs, potential revenue and itinerary-based passenger demand. Due to the large number of scheduled flights and the dependency of other airline operations (e.g., crew scheduling) on fleet assignment, solving the underlying *fleet assignment model* (FAM) is never an easy task. What further complicates the problem are:

- **Stochastic demand**: the fleet assignment decision is made 10 - 12 weeks in advance of departure, and at such an early stage, the level of market demand uncertainty is usually high. Deterministic models using average demand are thus inadequate to reflect the final demand realization.
- **Network effects**: to have a more accurate estimate of passenger flow and revenue, spill and recapture effects need to be properly addressed in an itinerary-based FAM. In reality, passengers spilled from an itinerary usually compare all the available options in the market and choose the most attractive one in terms of price, departure and arrival time, number of stops, total duration, and other factors that affect their preferences.

As a result, airlines usually solve an initial FAM based on early demand forecasts, and later swap aircraft assignments in response to demand variations (demand driven dispatch). In terms of spill and recapture, they either assume a constant recapture rate or penalize any spill or unmet capacity in the objective function.

In this paper, we present a two-stage stochastic model to explicitly consider potential market scenarios. The first stage is a basic fleet assignment model which decides the assignment of fleet types to flight legs. The second stage then finds the optimal passenger flow on available itineraries for each scenario based on the assigned capacities and the demand estimated for each itinerary. Our model inherits the attractiveness-based spill and recapture framework from Wang et al. (2014) which does not consider demand stochasticity, and by doing so, any spilled passenger is recaptured with a probability proportional to the attractiveness of an alternative itinerary. Another novelty of the proposed model is the integration with a production simulator which generates market scenarios and solves the corresponding second stage problems by considering bookings of multiple periods over the past year and advanced options such as up- and down-sells. This brings our model closer to a point of potential integration with airlines' decision support systems and making immediate impact. Unfortunately, due to the size of the model and the number of scenarios under consideration, solving it directly is computationally challenging. Hence, developing effective solution methodologies is also a focus of this paper.

Studies on fleet assignment can be traced back 30 years. Abara (1989) was one of the first researchers to examine realistically-sized FAM using a connection network. In contrast, Berge and Hopperstad (1993)

and Hane et al. (1995) were among the first researchers to use the time-space network, which has quickly become the dominant method to formulate FAM. A notable effort in modeling fleet assignment is the itinerary-based FAM proposed by Barnhart et al. (2002), which combines the basic fleet assignment model with a passenger mix model to explicitly capture network effects. Unlike our attractiveness-based approach, they assume that the rate of recapture is calculated through a Quantitative Share Index (QSI) and remains constant regardless of what options/itineraries are available in the market.

The works that are most closely related to ours are Sherali and Zhu (2008) and Listes and Dekker (2005). Sherali and Zhu (2008) propose a two-stage stochastic model in which the first stage conducts an initial fleet assignment by making only family-level assignments to flight legs, while the second stage performs subsequent family-based type-level assignments to accommodate passenger demand for each scenario. They conduct polyhedral analysis and develop a Benders decomposition-based solution approach. In our work, we do not consider the family-level assignment separately but complete all the assignment tasks in the first stage. The second stage of our model focuses on attractiveness-based network effects and is a more complicated production version that considers additional features such as up- and down-sells and fare estimation. In terms of the solution approach, we develop Benders decomposition-based algorithms tailored for a distributed computing environment to further reduce the computational time. Listes and Dekker (2005) also take the stochasticity of demand into consideration, but they focus on fleet composition[1] rather than fleet assignment and they do not consider spill and recapture. Their solution approach is based on progressive hedging (PH). Due to the size and complexity of individual scenarios, they pursue the strategy of first finding a solution to the linear relaxation of the model, and then using a simple rounding procedure to obtain integer results. In this paper, we also propose a heuristic algorithm based on PH, but what makes our work significantly different from theirs are: 1) we incorporate a Benders decomposition framework into the traditional PH; 2) we do not solve a linear relaxation of the model. To address the computational challenge, we employ a distributed framework named MapReduce. We decide to use MapReduce instead of other more traditional and in a certain sense more powerful architectures such as MPI because Hadoop and MapReduce are now ubiquitous in practice. The majority of companies have Hadoop installations and data scientists using them. This is definitely not the case for MPI.

The three algorithms developed in this paper are distributed versions of the classic Benders decomposition and PH algorithms. In the Benders decomposition-based approaches, all the scenario-dependent subproblems (i.e., the second stage of our model) are solved and the corresponding Benders optimality cuts are generated simultaneously on cluster computers to reduce the computational difficulty of having multiple scenarios. Our PH algorithm works similarly by solving the scenario problems in parallel. However, unlike the traditional PH approach, we solve each scenario problem by a Benders decomposition algorithm and add the Benders cuts generated in previous iterations as extra constraints to the next iteration. With such preservation of Benders cuts and the presence of binary variables in our model, this approach is only a heuristic. We evaluate and compare these algorithms based on a medium-size airline, and our computational results establish feasibility in using the two-stage stochastic model and the aforementioned distributed methodologies to solve FAM. A 10 - 15% increase in profitability is observed, and the optimality gaps from all the algorithms are less than 12%.

The contributions of this paper are as follows.

1. According to authors' best knowledge, this is the first effort to apply MapReduce to address the computational challenge of a large-scale two-stage stochastic program. An extension to multi-stage is straightforward.

2. This is the first time that the stochastic nature of passenger demand is explicitly addressed under an attractiveness-based spill and recapture framework. The two-stage stochastic model we formulate is an improvement over the deterministic version proposed in Wang et al. (2014).

3. The incorporation of a Benders decomposition algorithm into PH, and preserving Benders cuts from previous iterations as additional scenario problem constraints is a novel modification of the

[1]Fleet composition: given a set of aircraft types, the fleet composition problem is to determine the optimal number of aircraft of each type to be used to maximize the profit.

conventional progressive hedging approach.

4. We integrate the proposed algorithms with a production simulator developed by Sabre Airline Solutions to solve the second stage, and thus make them immediately applicable to industrial practitioners.

The rest of this paper is organized as follows. In Section 2, we formulate the two-stage stochastic model with attractiveness-based spill and recapture effects. Section 3 presents the three distributed algorithms based on MapReduce. Section 4 gives the findings from our computational experiments and presents sensitivity analyses. Finally, we conclude in Section 5.

2 A Stochastic Formulation of the Problem

In order to take the stochastic nature of passenger demand into explicit consideration, we formulate a two-stage stochastic model that considers several potential market scenarios. The entire formulation relies on the notion of an embedded time-space network, which is typically used in formulating FAM and representing flight schedule. To learn more about this network and its applications, we refer readers to, for example Sherali et al. (2006).

The first stage of our model assigns aircraft to the scheduled flight legs on a daily basis so that the following criteria are met: 1) Each flight leg is covered by exactly one fleet type; 2) Aircraft arriving at an airport at a particular time must leave that airport at some time later to ensure the same daily schedule and avoid deadheading; 3) Only available aircraft of each fleet type are used in the assignment. These criteria are usually referred to as *cover constraints*, *conservation of flow constraints* and *aircraft count constraints*. Since this part has been repeatedly formulated in the classic fleet assignment literature (e.g., Barnhart et al. (2002)), it is hence omitted in this paper. The second stage then utilizes the assigned capacity to accommodate the itinerary-based demand for each scenario, and adopts the attractiveness-based network model of Wang et al. (2014) to handle the spilled passengers. We incorporate this framework because:

1. Modeling spill and recapture is essential for passenger flow estimation and capacity planning, and hence relates to the fleet assignment problem when matching fleet types with different capacities to flight legs.

2. In reality, (spilled) passengers constantly change their preferences based on all the available itineraries in the market. This should be reflected in modeling.

The following notation is used. An itinerary always has the underlying market. Sometimes we show this relationship explicitly in order to stress it.

Sets:

- L: set of scheduled flight legs, indexed by l.
- K: set of fleet types, indexed by k.
- M: set of all markets, indexed by m.
- $\Pi_m^{HA}(l)$: set of all itineraries of the host airline (HA) that include flight leg l in market m, $m \in M$, $l \in L$.
- S: set of scenarios in terms of demand realization, indexed by s.

Decision Variables:

- $x_{lk} = 1$ if flight leg l is flown by fleet type k; 0 otherwise.

- q_i^s: market share of itinerary i in market m under scenario s, $i \in \Pi_m^{HA}$, $m \in M$, $s \in S$.
- q_m^s: market share of all the itineraries offered by other airlines in market m under scenario s, $m \in M$, $s \in S$.
- q^s: vector of market share over all markets under scenario s, $s \in S$. That is, $\big((q_i^s)_i, (q_m^s)_m\big)$.

Parameters:

- Cap_k: seat capacity of fleet type k, $k \in K$.
- c_{lk}: cost of assigning fleet type k to flight leg l, $k \in K$, $l \in L$.
- p^s: probability for realization of scenario s, $s \in S$.
- D_m^s: total demand realization for market m under scenario s, $m \in M$, $s \in S$.
- D^s: vector of demand realizations over all markets under scenario s, $s \in S$. That is, $D^s = (D_m^s)_m$.

The general form two-stage stochastic model reads as follows, where $x = (x_{lk})_{l,k}$.

$$\min \sum_{l \in L} \sum_{k \in K} c_{lk} x_{lk} - \sum_{s \in S} p^s Q(x, D^s) \tag{2.1}$$

$$\begin{aligned} \text{s.t.} \quad & \textit{Cover Constraints} \\ & \textit{Conservation of Flow Constraints} \\ & \textit{Aircraft Count Constraints} \end{aligned}$$

where, for each $s \in S$, we have

$$Q(x, D^s) = \max_{q^s} \; r(D^s, q^s) \tag{2.2}$$

$$\text{s.t.} \quad e(D^s, q^s) \leq \sum_{k \in K} x_{lk} Cap_k \qquad l \in L \tag{2.3}$$

$$q^s \in V \tag{2.4}$$

The objective function (2.1), together with (2.2), minimizes the total expected cost, or equivalently, maximizes the total expected profit. The first stage includes the typical cover constraints, flow balance constrains and aircraft count constraints to make the assignment of fleet types to flight legs. The second stage is a passenger mix model with attractiveness-based spill and recapture. The objective is to find the optimal passenger flow on all the available itineraries, subject to the limited capacity assigned to the flights and the demand estimated for each itinerary. Unlike Wang et al. (2014) or any traditional approaches, in this paper, we use a sophisticated production version of $Q_s(x, D^s)$ where bookings over a year are accumulated and advanced options such as up- and down-sells are captured. A simplistic model is given next.

2.1 A Simplistic Second Stage Model

The following additional notation is used in the simplistic model.

- f_i^s: average fare estimate for itinerary i under scenario s, $i \in \Pi_m^{HA}$, $s \in S$.
- A_i: attractiveness of itinerary i in market m, $i \in \Pi_m^{HA}$, $m \in M$. According to Wang et al. (2014), the attractiveness can be quantified by e^U, where utility U is a linear combination of a series of attributes such as departure time, arrival time, number of stops, total duration and so on.

· A_m: attractiveness of all the itineraries from other airlines in market m, $m \in M$. This represents passenger utility of flying with a different airline.

The model hence reads as:

$$Q(x, D^s) = \max_{q^s} \sum_{m \in M} D_m^s \sum_{i \in \Pi_m^{HA}} f_i^s q_i^s \tag{2.5}$$

$$\text{s.t.} \quad \sum_{m \in M} D_m^s \sum_{i \in \Pi_m^{HA}(l)} q_i^s \leq \sum_{k \in K} x_{lk} Cap_k \qquad l \in L \tag{2.6}$$

$$\sum_{i \in \Pi_m^{HA}} q_i^s + q_m^s = 1, \qquad m \in M \tag{2.7}$$

$$A_m q_i^s \leq A_i q_m^s, \qquad i \in \Pi_m^{HA},\ m \in M \tag{2.8}$$

$$q^s > 0 \tag{2.9}$$

Objective function (2.5) maximizes the revenue captured from all the itineraries of the host airline. Constraints (2.6) impose that the total demand captured on all itineraries that include leg l cannot exceed the capacity assigned to leg l. Constraints (2.7) and (2.8) ensure that the probability of recapturing the spilled demand is proportional to the attractiveness of the alternative itinerary. This model modifies the work of Wang et al. (2014), and is only used to explain the concept of attractiveness-based network effect.

3 Algorithms

The two-stage stochastic model developed in Section 2 has mixed binary variables in the first stage and continuous variables in the second stage. Since the number of daily (or weekly) flights/itineraries for a major airline can easily reach several thousands, and the proposed model also explicitly considers multiple scenarios, it turns out to be a very hard problem. Efficient solution methodologies hence need to be developed to mitigate the inherent combinatorial burden and to supply a practically good solution within a reasonable computational time.

In this section, we first introduce a decision support simulator named Airline Planning and Operations Simulator (APOS), and explain how to use it to generate random demand realizations and get $Q(x, D^s)$. We then present three distributed algorithms - two Benders decomposition-based algorithms and one progressive hedging-based algorithm - to solve multiple scenarios concurrently to reduce the overall computational time. All the distributed algorithms developed in this paper are based on the MapReduce framework.

3.1 Integration of APOS

We start by formally introducing the Airline Planning and Operations Simulator (APOS). Airlines use various methodologies in their planning processes. The most reliable and widely used method to assist decision making is simulation that allows to replicate actual customer behavior in the controlled environment and to carry out multiple experiments on the same set of inputs. APOS, provided by Sabre Airline Solutions, was originally designed to fulfill these tasks for the sake of revenue management. In particular, it reads historical customer booking information to generate forecast streams, which we call potential demand realizations in the context of this paper, and then performs optimization algorithms on the generated streams to achieve the highest expected revenue. With slight modification, the simulator is able to provide the following two crucial functions which will be repeatedly used in designing our overall algorithms.

1. Given a random seed, APOS generates a particular market level demand realization or scenario.

2. Given a feasible fleet assignment solution (i.e., a feasible first stage solution), APOS solves the attractiveness-based passenger mix model in the second stage and returns the highest revenue, $Q(x, D^s)$.[2] Meanwhile, it also generates other outputs, such as the dual solution, to reveal more information about the second stage model it solves.

Therefore, as shown in Figure 1, we incorporate APOS into the two-stage stochastic model as a blackbox function to generate multiple scenarios and solve the corresponding second stage problem (2.2) - (2.4) for any given scenario. Being a blackbox, how the simulator actually operates and what algorithms it employs are out of the scope of this paper, and thus omitted.

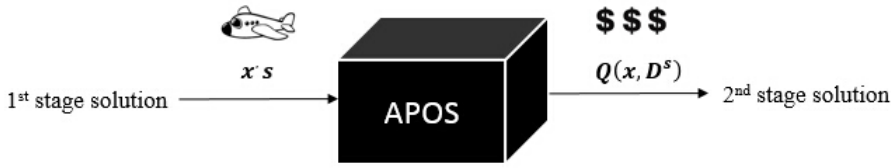


Figure 1: Integration of APOS as a blackbox

3.2 Three MapReduce-based Decomposition Algorithms

In this section, we propose three decomposition algorithms to partition the problem into multiple smaller subproblems - two primal decomposition methods based on Benders decomposition and one dual decomposition method based on progressive hedging. Since all these algorithms involve solving multiple independent subproblems with similar structures, parallel computing has great potential to reduce the computational time by solving the subproblems simultaneously. Thus, all the algorithms to be exhibited in this section are developed based on the distributed framework named MapReduce.

MapReduce is a software framework to submit and process parallelizable applications which deal with vast amount of data on clusters of computers. In this paper however, we leverage this popular "big data" technology to solve a large-scale stochastic program. A MapReduce program is typically composed of **mappers** which turn each data record into a key-value pair and **reducers** which perform a summary operation. In our case, a mapper instructs a number of reducers which scenario to solve, and then each reducer calls APOS to solve the second stage model of the assigned scenario. Further details about the implementation are listed in Section 4.1. The input to the algorithm includes a list of scenario ID's (e.g., $1, ..., |S|$) and other standard input to FAM.

3.2.1 Two MapReduce-based Benders Decomposition Algorithms

Benders decomposition is a popular method for solving certain large-scale optimization problems, such as the two-stage stochastic programming model formulated in this paper. The solution process of a typical Benders algorithm alternates between a master problem and subproblems corresponding to different scenarios. In our case, the master problem is solved by branch-and-bound in the space of the first stage variables, and each subproblem is solved by executing APOS with a constant seed specifying the scenario. The corresponding Benders optimality cut is then constructed by utilizing the dual information as the classic Benders algorithm of Benders (1962) and Kalvelagen (2002). We omit the details of the classic Benders algorithm since it has been well documented in a variety of literature.

In our MapReduce-based algorithms, all the scenario-dependent subproblems are solved in parallel and the associated Benders cuts are generated separately. As we can observe from Figures 2 and 3, with all these cuts available, the Benders algorithm can proceed with either of the following two directions: 1) adding all the cuts to the master problem and repeat the solution process until some stopping criterion

[2]APOS offers a function mode named *SBLP*. It is developed based on the passenger mix model of Wang et al. (2014), but also incorporates many practical considerations (up- and down-sells, cargo capacity, etc).

is met; 2) aggregating all these cuts into a single cut by using the expected value of coefficients and then adding it to the master problem to continue the solution process. The following Algorithms 1 and 2 are hence developed in line with these two directions.

Algorithm 1 The MapReduce-based Multiple-cuts Benders Algorithm

1: Initialization:
   $LB := -\infty$, $UB := \infty$
2: Solve the first stage master problem
3: Update LB
4: **while** stopping criteria not met (e.g., $UB - LB > \epsilon$) **do**
5: Call MapReduce to solve each second stage subproblem and generate Benders cuts $C_1, ..., C_{|S|}$
   Mapper: Assign one scenario ID to each reducer
   Reducer: Run APOS to solve the second stage of the assigned scenario $s \in S$, and generate C_s
6: Retrieve $C_1, ..., C_{|S|}$
7: Update UB
8: Add $C_1, ..., C_{|S|}$ to the master problem
9: Solve the first stage master problem
10: Update LB
11: **end while**

Algorithm 2 The MapReduce-based Aggregate-cut Benders Algorithm

1: Algorithm 1 steps 1 - 7
2: Aggregate $C_1, ..., C_{|S|}$ into a single cut $\bar{C}$ using the expected coefficients
3: Add $\bar{C}$ to the master problem
4: Algorithm 1 steps 9 - 11

In the above algorithms, the parallelization is done through the MapReduce framework and APOS is treated as an integrated function, which can be repeatedly called in the reducing phase. The MapReduce framework consists of mappers and reducers, in which mappers take the scenario ID's (i.e., $1, ..., |S|$) as input and emit a unique ID to each reducer, and each reducer takes the solution of the master problem as input via distributed cache (see Section 4.1 for details) and executes APOS to solve the second stage subproblem based on the ID assigned to it. The two algorithms only differ in the way of handling Benders cuts and thus have the same complexity.

3.2.2 A Progressive Hedging Algorithm based on MapReduce

When confronted with a two-stage stochastic program for which there exist effective algorithms to solve individual scenarios, the progressive hedging (PH) method is another popular approach. However, in the context of our model, it is not possible to explicitly solve the (deterministic) problem corresponding to each scenario because APOS is considered a blackbox and thus we do not know how it generates the scenario and what model and algorithms it employs. To get around this difficulty, a natural starting point is to construct a Benders decomposition framework similar to what we developed in Section 3.2.1 so that the second stage of each given scenario can be solved by APOS without knowing the model or the solution process. That being said, in each iteration of the to-be-proposed PH algorithm, Benders decomposition is employed for solving scenario problems.

A MapReduce-based approach is again proposed to solve the independent scenario problems in parallel. Mappers still read the scenario ID's as input and then feed each reducer a unique ID. However, for any given scenario $s \in S$, the reducer no longer only solves the second stage problem and generates a single cut, but uses Benders decomposition to iteratively solve the complete model specified by scenario s. That being said, if we let P_1^s and P_2^s be the first and second stages of the scenario problem P^s, where $s \in S$, the output from step 5 of Algorithms 1 and 2 is a Benders optimality cut corresponding to P_2^s while

the output from the same step of the following PH algorithm is a solution to P^s. The PH algorithm is essentially a sequential version of Algorithms 1 and 2 performed on individual scenarios.

A potential issue with the above solution approach is that each scenario problem, although much smaller than the overall problem, is still not easy to solve to optimality in a relatively short time. To address this challenge, we modify the PH algorithm in such a way that the integrated Benders solution process in every PH iteration is terminated earlier before an optimal solution is found and all the Benders cuts generated from previous iterations are retained and added to the next iteration. It is easy to see that these cuts are valid since they are based on the same scenarios with only objective function being varied. This leads to the following Algorithm 3, taking ρ as a penalty factor.

Algorithm 3 The MapReduce-based Progressive Hedging Algorithm

1: Set $v := 0$
2: Initialize $\bar{x}^{(0)}$ and $(w^s)^{(0)}$
3: **while** stopping criteria not met **do**
4: Set $v := v + 1$
5: Call MapReduce to solve each scenario problem in parallel and obtain $(x^s)^{(v)}$
Mapper: Assign one scenario ID to each reducer
Reducer: Use the Benders decomposition algorithm to solve the assigned scenario $s \in S$:

$$(x^s)^{(v)} := \arg\min_x \left(\sum_{l\in L}\sum_{k\in K} c_{lk}x_{lk} - Q(x, D^s) + \sum_{l\in L}\sum_{k\in K} (w^s_{lk})^{(v-1)}x_{lk} + \frac{\rho}{2}\sum_{l\in L}\sum_{k\in K} \|x_{lk} - \bar{x}^{(v-1)}_{lk}\|^2 \right)$$

s.t. 1^{st}- and 2^{nd}-stage constraints and Benders cut constraints from iterations $v-1, v-2, ..., 0$
6: Retrieve $(x^s)^{(v)}$ and Benders cuts from each scenario
7: Set

$$\bar{x}^{(v)}_{lk} := \sum_{s\in S} p^s (x^s_{lk})^{(v)}, \qquad (w^s_{lk})^{(v)} := (w^s_{lk})^{(v-1)} + \rho\big((x^s_{lk})^{(v)} - \bar{x}^{(v)}_{lk}\big)$$

8: Add retrieved Benders cuts to corresponding scenario problems
9: Substitute each $((x^s)^{(v)})$ into (2.1), and set

$$x^{(v)} := \arg\min_s \sum_{l\in L}\sum_{k\in K} c_{lk}(x^s_{lk})^{(v)} - \sum_{s\in S} p^s Q\big((x^s)^{(v)}, D^s\big)$$

10: **end while**

In the above algorithm, since all the scenario-dependent solutions from Step 6 are feasible, we substitute each of them into (2.1) for cost evaluation and pick the one with the minimal cost as an overall primal solution. We also tried other more sophisticated ways of getting an overall solution, for example, solving $x^{(v)} := \arg\min_s \sum_{l\in L}\sum_{k\in K} -\omega_{lk}x_{lk}$ subject to the cover constraints, conservation of flow constraints and aircraft count constraints. The weight ω_{lk}, $l \in L$, $k \in K$ is determined by how many times fleet type k is assigned to flight leg l across all scenario solutions. The more often k is assigned to l, the larger the weight would be, thus leading to a smaller coefficient in the above expression and a higher chance of $x_{lk} = 1$ in the overall solution. In this paper, we go with the simple approach in Algorithm 3 as it gives the best solution. Graphical illustrations of the three algorithms are given in Figures 2 and 3.

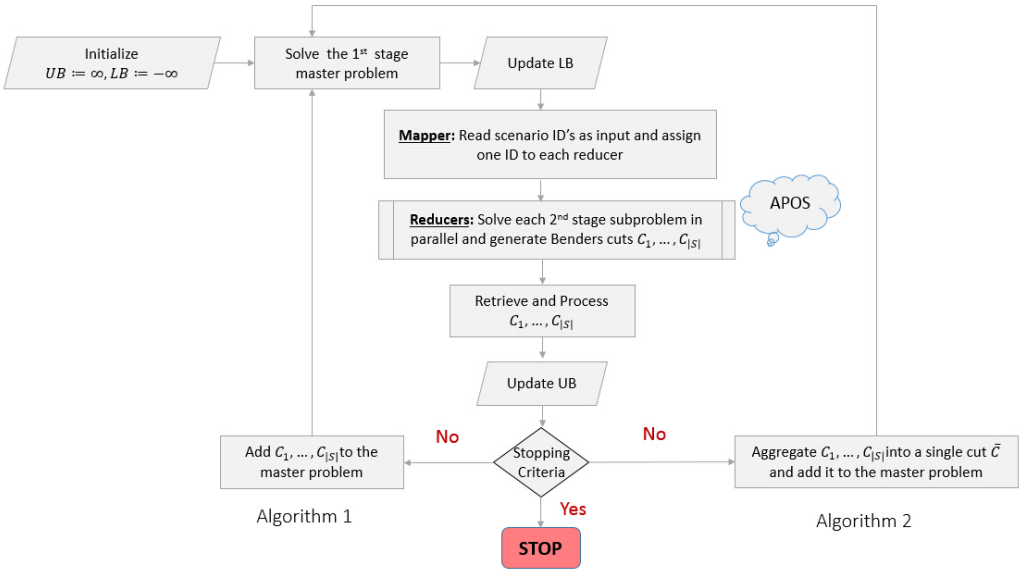


Figure 2: Solution process flow of the two MapReduce-based Benders decomposition algorithms

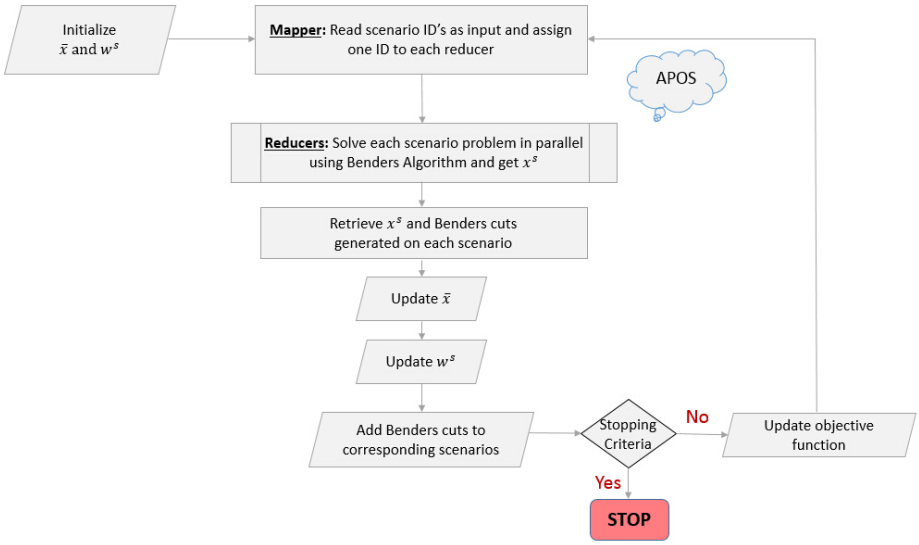


Figure 3: Solution process flow of the MapReduce-based PH algorithm

4 Computational Experiments

4.1 Implementation

All computational experiments are conducted on a Hadoop cluster. The cluster consists of four 2.2GHz Xeon CPU's, each having 8 cores and hence resulting in 32 cores for parallel program execution. Each of the four servers has 32 GBytes of main memory. They serve as data nodes and there is a separate named node. Cloudera version 5 is the backbone Hadoop installation. All the optimization problems are solved by IBM ILOG CPLEX Optimization Studio.

To test the performance of the proposed algorithms, we use data from a medium-size airline which involves 132 international flights and 20 fleet types. Since we have 32 cores, 32 scenarios are generated by APOS to reflect market demand fluctuations. The input to the simulator includes information of equipments (fleet types), flights, markets and days when booking information is updated and the forecast is reoptimized. In our experiments, these days are specified as 366, 226, 107, 57, 35, 23, 17, 13, 9, 5, 3 and 1, so that bookings of various time periods over the past year are accumulated to capture the cyclic nature of passenger demand. All the data is provided by Sabre Airline Solutions, and the utility coefficients in attractiveness are calibrated by APOS.

By means of extensive numerical experiments, we notice that 5 hours of running time is enough to get a good quality solution from the proposed algorithms, and running additional hours does not lead to obvious further convergence. Therefore, for the rest of this section, running time not exceeding 5 hours is imposed as a stopping criterion for Algorithms 1 - 3. The resulting optimality gaps are all around 10%. We also set the relative MIP gap tolerance (a Cplex parameter) to 10^{-2} to solve the first stage master problem faster[3]. Since all three algorithms involve solving a Benders master problem somehow, this setup is crucial in terms of expediting the overall solution approaches. Finally, instead of seeking an optimal solution in every iteration of Algorithm 3, we stop the solution process of each scenario problem after 15 minutes of execution or after 20 cuts are generated. The motivation of doing this is because: 1) with the presence of binary variables, our PH approach is only a heuristic, and thus an optimal solution in each iteration does not guarantee any overall convergence; 2) 15 minutes of execution or 20 Benders iterations usually leads to a reasonably good solution for a single scenario problem, with the corresponding optimality gap around 30%.

In Algorithms 1 - 3, both mappers and reducers are implemented in Python using Hadoop Python streaming. Since Hadoop Python API does not support user-defined partitioning, to ensure each reducer only processes one scenario, we mimic the partitioning function by: 1) randomly generating 32 java string hashcodes with equal length; 2) using them as scenario IDs. As a result, the default partitioner, which partitions the data using Java string hashcode as hash key, splits the output from the mappers into 32 segments (scenarios) and gives them to 32 reducers.

From the implementation standpoint, passing information to or retrieving information from MapReduce is not straightforward and hence needs further discussion. There is an overarching external scripts that executes the highest level loop. In each outer iteration of Algorithms 1 and 2, we pass the first stage master problem solution to each reducer using Hadoop Distributed Cache. Specifically, in each iteration we write the solution to a text file and then cache it so that it is accessible to all the data nodes (reducers). Once reducers finish solving all the subproblems, information to construct Benders optimality cuts (i.e., dual information, coefficient matrices and right-hand sides) is written to the Hadoop Distributed File System (HDFS). To retrieve these cuts and add them to the master problem for next iteration, the external script copies the output of reducers from HDFS to the local filesystem for further processing. In Algorithm 3, Benders cuts and $(x^s)^{(v)}$ corresponding to each scenario are retrieved in a similar way through HDFS. However, the information passed to the reducers is significantly different since each reducer no longer only solves the second stage subproblem and generates a single cut, but employs Benders decomposition to solve a complete model (with both first and second stages). For this reason, in every iteration of Algorithm 3, each reducer requires a scenario-dependent master problem and the additional Benders cut constraints. Information to construct these models is written to a text file and sent as distributed cache to make it accessible to all the reducers. No LP or MPS files are transferred in the above implementation: only customized text files are used. Finally, APOS is a C++ library which is called from a reducer through the standard Python-to-C++ interfacing.

4.2 Computational Results

In order to examine the benefits of the proposed algorithms, we take the airline's current fleeting decision as a benchmark solution. The most important computational results are presented in Table 1, in which a positive sign indicates an increase in the objective value or a decrease in profit, and a negative sign

[3]The default setting of the MIP tolerance gap in Cplex is 10^{-4}.

indicates the opposite. These principles are followed by all the tables and figures presented in this section. All solutions including the benchmark solution are compared with respect to the expected cost based on the 32 generated scenarios. As we can see, the solutions from the proposed algorithms all yield cost reduction against the benchmark, and a 10 - 15% improvement is often considered significant for airlines with tight margins. The two Benders decomposition-based algorithms output similar objective values and optimality gaps. The solution approach with 32 Benders cuts added to the master problem in each iteration (Algorithm 1) gives a smaller objective value, while the approach adding a single aggregate cut (Algorithm 2) renders a narrower optimality gap. Between these two, we also see more Benders iterations in Algorithm 2, which is intuitive since we add fewer constraints to the master problem in every iteration and that usually leads to a simpler problem and a faster solution process. Algorithm 3 yields the best solution in Table 1. As shown in Figure 4, a solution superior to those given by the Benders algorithms can be achieved within 3 hours. We also observe a large discrepancy in the objective values favoring Algorithm 3 in the first two hours of execution. Thus, the superiority of this Benders-incorporated PH approach is demonstrated and it is recommended when a quick solution is desired.

Table 1: Comparison of the three MapReduce-based algorithms

	Obj. vs. Benchmark$_{32}$	Optimality Gap	# of Benders Iterations
Algorithm 1	-11.5%	11.9%	86
Algorithm 2	-10.2%	10.2%	189
Algorithm 3	-12.4%	—	5,465

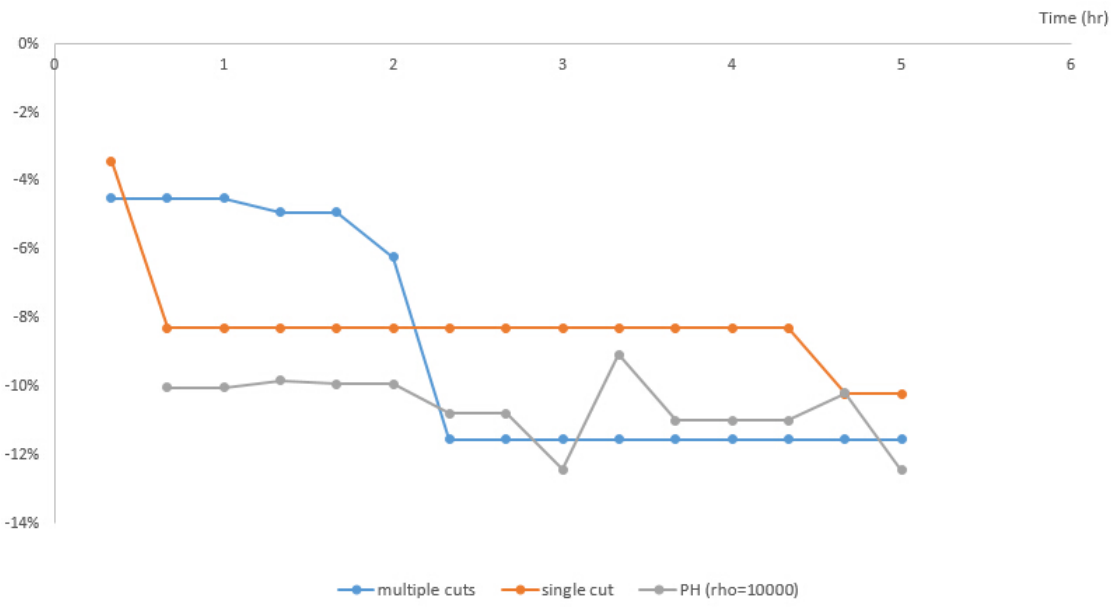


Figure 4: Obj. vs. Benchmark$_{32}$ in 5 hours

Due to the the presence of binary variables in our model, Algorithm 3 is simply a heuristic and the optimality gap is theoretically intractable. Furthermore, since we conduct 32 independent Benders decomposition processes in each iteration of Algorithm 3, a large number of Benders iterations (i.e., over 5,000) is expected.

Next, we evaluate both the benchmark solution and the solutions from the three algorithms on 1,000 scenarios for a more accurate cost estimate. The evaluation process is simple: given a solution x_{lk}, $l \in L$, $k \in K$, we are able to calculate $\sum_{l \in L} \sum_{k \in K} c_{lk} x_{lk}$ in (2.1); then we pass x_{lk} and 1,000 distinct scenario IDs

sequentially to APOS to let it solve for each $Q_s(x, D^s)$, $s \in S$ with $|S| = 1,000$. With p^s being a known parameter, we can easily get the objective values corresponding to each solution. The results are given in Table 2, where *Obj.* is obtained based on 32 scenarios and *Eval.* is evaluated based on 1,000 scenarios. As we can see, the solutions from the proposed algorithms still yield improvement over the benchmark, and Algorithm 3 still outputs the best result, followed by Algorithm 1 and then Algorithm 2. We also notice that all three objective values increase compared to those evaluated based on 32 scenarios. This is intuitive since our solutions are derived by solving the two-stage stochastic program with 32 scenarios, they are unlikely to be optimal under the additional scenarios. Therefore, the percentage increase shown in the second column of Table 2 can be considered as a measure of the robustness of a solution. Following this argument, the solution from the PH approach appears to be the most robust one.

Table 2: Evaluation of the solutions on 1,000 scenarios

	Eval. vs. Benchmark$_{1,000}$	Eval. vs. Obj.
Algorithm 1	-9.8%	+11.2%
Algorithm 2	-1.3%	+17.1%
Algorithm 3	-22.6%	+1.6%

4.2.1 Sensitivity Analysis on Cost Coefficients

We perform sensitivity analysis to examine how well the proposed algorithms work when operating cost[4] increases (or decreases) by 5 or 10%. The computational results are presented in Table 3. All the evaluations are based on the 32 generated scenarios. As we can see, the solutions given by the proposed algorithms are still 10 - 15% better than the benchmark however the operating cost varies. The only exception happens when cost increases by 5%, in which case Algorithm 1 yields only a 1.2% improvement over the benchmark while Algorithm 3 gives a surprisingly high 23.3%. Algorithm 3 seems to dominate the other two in all cases, and from Figure 5, we can make a similar argument as in Section 4.2 that it is able to reach a better solution faster. However, this dominant performance of Algorithm 3 does not imply that the two Benders decomposition-based algorithms are no longer necessary. In fact, if a longer execution time is available and an optimal solution is desired, we still need the two Benders algorithms since Algorithm 3 is only a heuristic without any convergence guaranteed.[5]

From Table 3, we also see that the profit of the airline is very sensitive to the operating cost fluctuations. For example, a 5% decrease of operating cost leads to an 80% increase in profit, while a 5% increase reduces the profit by more than 85%. This proves the tight margin of airlines and the importance of cutting costs by utilizing equipment and labor more efficiently.

Table 3: Sensitivity analysis on cost coefficients

	Obj. vs. Benchmark$_{32}$				Obj. Changes			
Cost Coef	$\downarrow 10\%$	$\downarrow 5\%$	$\uparrow 5\%$	$\uparrow 10\%$	$\downarrow 10\%$	$\downarrow 5\%$	$\uparrow 5\%$	$\uparrow 10\%$
Algorithm 1	-11.6%	-11.6%	-1.2%	-11.0%	-165.7%	-79.6%	+88.1%	+171.1%
Algorithm 2	-11.2%	-12.4%	-9.8%	-10.8%	-168.1%	-83.2%	+86.9%	+172.2%
Algorithm 3	-13.1%	-14.7%	-23.3%	-12.3%	-167.3%	-83.2%	+85.6%	+172.4%

4.2.2 Sensitivity Analysis on Number of Scenarios

Besides operating cost, how the number of scenarios impacts the final solution is also worth of exploring because if considering more scenarios does not give a better solution, we rather consider a simpler model with fewer scenarios to expedite the solution process. Therefore, in this section, we perform the proposed

[4]Operating cost refers to the cost coefficients in (2.1).

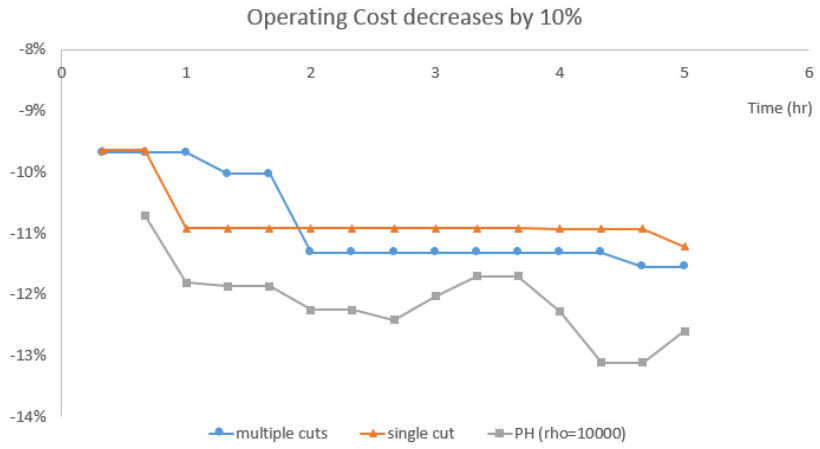
[5]Further experiments show that Algorithms 1 and 2 would prevail after 15 hours of running time.

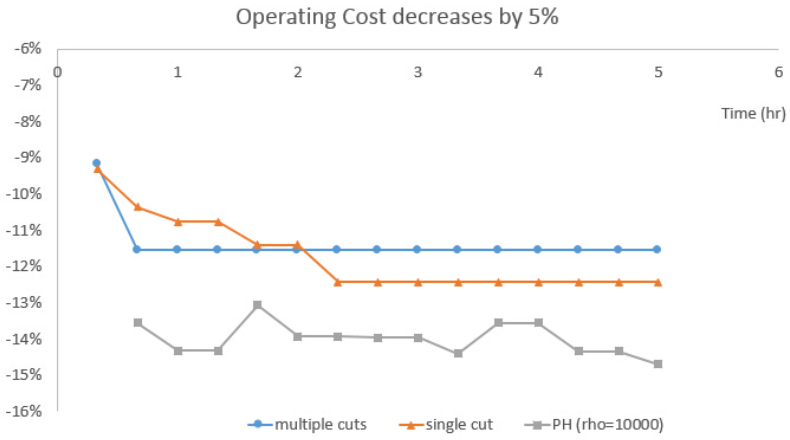
algorithms on 8 and 16 scenarios, and compare the solutions to the ones based on 32 scenarios. To see how these solutions work in reality (with high uncertainty), they are compared with respect to the expected cost based on 1,000 scenarios.

Table 4: Sensitivity analysis on number of scenarios

	Eval. Changes		
	8 Scenarios	16 Scenarios	32 Scenarios
Algorithm 1	+6.3%	+3.5%	0
Algorithm 2	+1.9%	+3.5%	0
Algorithm 3	+1.3%	+0.5%	0

As indicated in Table 4, fewer scenarios lead to worse solutions among all three algorithms. This makes intuitive sense as the more scenarios we consider, the more likely our model is able to capture the real demand fluctuation and hence give a solution working robustly under the level of uncertainty possessed by 1,000 scenarios. An exception happens for Algorithm 2, in which case 8 scenarios perform better than 16 scenarios.





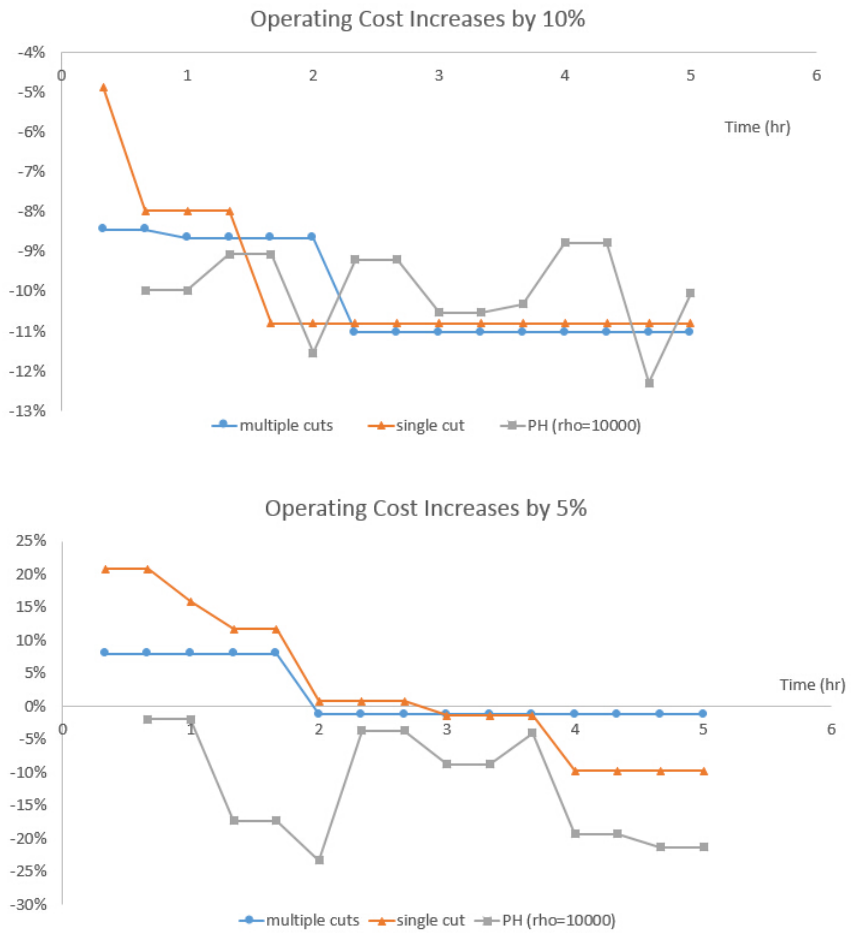


Figure 5: Obj. vs. Benchmark$_{32}$ in 5 hours when operating cost changes

5 Summary and Future Research

In this paper, we have proposed a two-stage stochastic fleet assignment model that considers potential demand scenarios and attractiveness-based spill and recapture effects. Most importantly, we developed three MapReduce-based solution approaches to solve each scenario-dependent subproblem in parallel to reduce the computational time. To examine the efficacy of the modeling and algorithmic strategies, computational results using real data from a medium-size airline were presented. It would be of interest of a follow-up paper to apply the proposed distributed framework to other stochastic problems, such as crew scheduling and financial portfolio management. Further computational experiments should be potentially conducted to examine if there exists an upper bound on the number of scenarios; once exceeding this value considering more scenarios would not lead to any significant changes in the final solution. And further research is necessary to incorporate additional features such as re-fleeting, flexible flight times (departure and arrival time windows) and schedule balance into our model. From the implementation standpoint, transferring the implementation to Apache Spark is also worthwhile attempting.

References

J. Abara. Applying integer linear programming to the fleet assignment problem. *Interfaces*, 19(4):20–28, 1989.

C. Barnhart, T. S. Kniker, and M. Lohatepanont. Itinerary-based airline fleet assignment. *Transportation Science*, 36(2):199–217, 2002.

J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, 1962.

M. E. Berge and C. A. Hopperstad. Demand driven dispatch: A method for dynamic aircraft capacity assignment, models and algorithms. *Operations Research*, 41(1):153–168, 1993.

Y. Fan and C. Liu. Solving stochastic transportation network protection problems using the progressive hedging-based method. *Networks and Spatial Economics*, 10(2):193–208, 2010.

C. A. Hane, C. Barnhart, E. L. Johnson, R. E. Marsten, G. L. Nemhauser, and G. Sigismondi. The fleet assignment problem: solving a large-scale integer program. *Mathematical Programming*, 70(1-3): 211–232, 1995.

E. Kalvelagen. Benders decomposition with GAMS, 2002. `http://www.amsterdamoptimization.com/pdf/stochbenders.pdf`.

O. Listes and R. Dekker. A scenario aggregation–based approach for determining a robust airline fleet composition for dynamic capacity allocation. *Transportation Science*, 39(3):367–382, 2005.

H. D. Sherali and B. M. P. Fraticelli. A modification of Benders' decomposition algorithm for discrete subproblems: An approach for stochastic programs with integer recourse. *Journal of Global Optimization*, 22(1-4):319–342, 2002.

H. D. Sherali and X. Zhu. Two-stage fleet assignment model considering stochastic passenger demands. *Operations Research*, 56(2):383–399, 2008.

H. D. Sherali, E. K. Bish, and X. Zhu. Airline fleet assignment concepts, models, and algorithms. *European Journal of Operational Research*, 172(1):1–30, 2006.

H. D. Sherali, K. H. Bae, and M. Haouari. Integrated airline schedule design and fleet assignment: polyhedral analysis and Benders' decomposition approach. *INFORMS Journal on Computing*, 22(4): 500–513, 2010.

B. C. Smith. *Robust airline fleet assignment*. PhD thesis, Georgia Institute of Technology, 2004.

B. C. Smith and E. L. Johnson. Robust airline fleet assignment: Imposing station purity using station decomposition. *Transportation Science*, 40(4):497–516, 2006.

H. Topaloglu and W. B. Powell. Dynamic programming approximations for stochastic time-staged integer multicommodity flow problems. *INFORMS Journal on Computing*, 18(1):31–42, 2006.

D. Wang, D. Klabjan, and S. Shebalov. Attractiveness-based airline network models with embedded spill and recapture. *Journal of Airline and Airport Management*, 4(1):1–25, 2014.

J. P. Watson and D. L. Woodruff. Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. *Computational Management Science*, 8(4):355–370, 2011.

X. Zhu. *Discrete two-stage stochastic mixed-integer programs with applications to airline fleet assignment and workforce planning problems*. PhD thesis, Virginia Tech, 2006.