Large-Scale Data-Driven Airline Market Influence Maximization

Duanshun Li*
duanshun@ualberta.ca
University of Alberta
Edmonton, Alberta, Canada

Jing Liu*
jliu11.job@gmail.com
Walmart Research Lab.
Reston, VA, USA

Jinsung Jeon
jjsjjs0902@yonsei.ac.kr
Yonsei University
Seoul, South Korea

Seoyoung Hong
seoyoungh.kr@gmail.com
Yonsei University
Seoul, South Korea

Thai Le, Dongwon Lee
{thaile,dongwon}@psu.edu
Penn State University
University Park, PA, USA

Noseong Park
noseong@yonsei.ac.kr
Yonsei University
Seoul, South Korea

ABSTRACT

We present a prediction-driven optimization framework to maximize the market influence in the US domestic air passenger transportation market by adjusting flight frequencies. At the lower level, our neural networks consider a wide variety of features, such as classical air carrier performance features and transportation network features, to predict the market influence. On top of the prediction models, we define a budget-constrained flight frequency optimization problem to maximize the market influence over 2,262 routes. This problem falls into the category of the non-linear optimization problem, which cannot be solved exactly by conventional methods. To this end, we present a novel adaptive gradient ascent (AGA) method. Our prediction models show two to eleven times better accuracy in terms of the median root-mean-square error (RMSE) over baselines. In addition, our AGA optimization method runs 690 times faster with a better optimization result (in one of our largest scale experiments) than a greedy algorithm.

CCS CONCEPTS

• **Theory of computation** → **Mathematical optimization**; • **Computing methodologies** → *Neural networks.*

KEYWORDS

large-scale optimization; transportation; deep learning

ACM Reference Format:
Duanshun Li, Jing Liu, Jinsung Jeon, Seoyoung Hong, Thai Le, Dongwon Lee, and Noseong Park. 2021. Large-Scale Data-Driven Airline Market Influence Maximization. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467423

*Duanshun Li and Jing Liu equally contributed to this work and were advised by Noseong Park when he was in George Mason University, Fairfax, VA, USA.

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467423

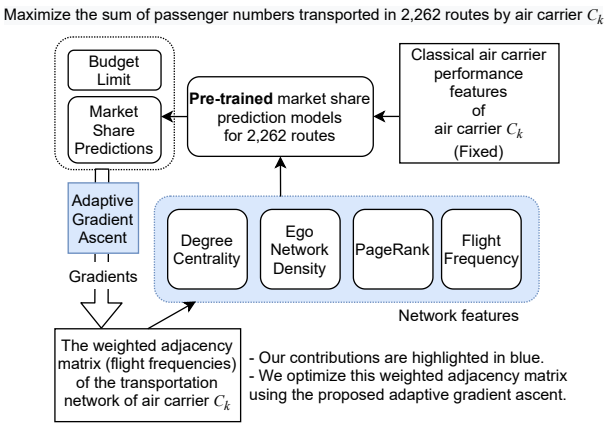


Figure 1: The architecture of the proposed prediction-driven optimization framework to maximize the market influence of air carrier C_k. Note that the pre-trained neural network-based market share prediction models constitute the objective function. The gradients of the budget constraint and the objective function can flow from the top to the bottom to optimize the weighted adjacency matrix because all intermediate modules are differentiable.

1 INTRODUCTION

Ever since the deregulation in 1978, there has been huge competition among US air carriers (airlines) for air passenger transportation. 771 million passengers were transported in 2018 alone and the largest air carrier produces a revenue of more than 43 billion dollars for the period between September 2017 and September 2018[1]. It is one of the largest domestic markets in the world and there is a huge demand to improve their services. Consequently, many computational methods have also been proposed to predict market share, ticket price, demand, etc. and allocate resources (e.g., aircraft) on those air passenger markets accordingly [3, 5, 6, 12].

The market influence is sometimes strategically more important than profits. Typically, there are two ways to expand business: i) a strategical merger with other strong competitors, and ii) a strategical play to maximize the market influence [11]. Our paper is closely related to the latter strategy.

[1]https://www.transtats.bts.gov

We propose a novel way of *unifying both data mining and mathematical optimization methods* to maximize air carrier's influence on the air transportation market. In this paper, we define the influence of an air carrier as *the number of passengers transported by the air carrier* which can be calculated by the total demand multiplied with the air carrier's market share.

Since the market influence of an air carrier in a route can be calculated by the total demand (passenger numbers) in the route multiplied with the market share, predicting market share is a key step in our work. Conventional features (e.g., average ticket price, flight frequency, and on-time performance) have been widely used to predict the market share [5, 6, 19, 20]. For instance, air carrier's market share on a route will increase if ticket price is decreased and flight frequency is increased. However, some researchers recently paid an attention to air carrier's transportation network connectivity that is highly likely to be connected to market share [17, 18]. As a response, we design a neural network-based prediction model that uses a wide variety of conventional and transportation network features, such as degree centrality, PageRank, and so forth. It is worth mentioning that we train a prediction model for each route.

On top of the market share prediction models, we build a budget-constrained optimization module to maximize the market influence by optimizing transportation network (more precisely, flight frequency values over 2,262 routes), which is an Integer Knapsack problem (cf. Fig. 1). Our objective function consists of the market share prediction models in those routes and our constraint is a budget limit of an air carrier. The objective is not in a simple form but rather a complex one of inter-correlated neural networks because changing frequency in a route will influence market shares on other neighboring routes as well. Therefore, it is very hard to solve with existing techniques that assume routes are independent from each other (see discussions in Section 2.2).

We test our optimization framework with 2,262 routes. To achieve such a high scalability, we design a method of **A**daptive **G**radient **A**scent (AGA). In our experiments, the proposed optimization method solves the very large-scale optimization problem much faster than existing algorithms. However, one main challenge in our approach is how to consider the budget constraint in the proposed gradient-based optimization technique — each air carrier has a limited budget to operate flights. It is not straightforward to consider the budget constraint with gradient-based optimization methods. However, our proposed AGA method is able to dynamically manipulate gradients to ensure the budget limit, i.e., dynamically impose a large penalty, if any cost overrun, in such a way that one gradient ascent update theoretically guarantees a decrease in the total cost. Therefore, a series of updates can eventually address the cost overrun problem.

In our experiments, our customized prediction model shows much better accuracy in many routes than existing methods. In particular, our median root-mean-square error is more than two times better than the best baseline. Our proposed AGA method is able to maximize the market influence on all those routes 690 times faster with a better optimized influence than a greedy algorithm.

2 RELATED WORK

We introduce a selected set of related works about air market predictions and optimizations.

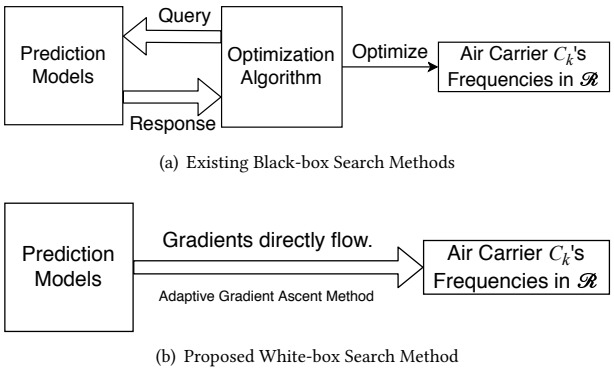


Figure 2: The comparison with existing black-box search methods in [5, 6] and the white-box search method proposed in this work. Our AGA optimization algorithm enables the white-box search concept to be used in this work.

2.1 Market Share Prediction

There have been proposed many prediction models such as [5, 6, 19, 20], to name a few. However, they share many common design points. First, almost all of them use the multi-logit regression model. It is a standard model to predict air transportation market shares. We also use the same multi-logit regression (see Section 3.1 for its details) after some extensions. Suzuki considers air carriers' frequency, delay, and safety [19] whereas Wei et al. study about the effect of aircraft size and seat availability on market share and consider other variables such as price and frequency [20]. There are some more similar works [5, 6]. In our paper, we consider transportation network features in addition to those conventional air carrier performance features.

2.2 Flight Frequency Optimization

One similar flight frequency optimization problem to maximize profits was solved in [5, 6]. In their work, An et al. showed that the frequency-market share curve is very hard to approximate with existing approximation methods such as piece-wise linear approximation [4]. After that, they designed one heuristic-based algorithm, called GroupGreedy, which runs an exact algorithm in each subset of routes (because running the exact algorithm for the entire route set is prohibitive). Each subset consists of a few routes and running the exact algorithm within a small subset provides a tolerable degree of scalability in general. However, they were able to test with *at most about 30 routes* for its prohibitively long execution time even with GroupGreedy and its scalability is not satisfactory. We test with 2,262 routes in this paper — i.e., the problem search space size is $O(n^{30})$ in their work vs. $O(n^{2,262})$ in this work.

In addition, we found that GroupGreedy cannot be used for our prediction model because of the network features — An et al. did not consider network features and assumed each route is independent [5, 6]. After adopting the assumption, they optimize for each route separately. In reality, however, changing a frequency in a route is likely to influence the market shares in other routes because routes are often inter-correlated. Thus, GroupGreedy based on the

Table 1: Comparison table between two related papers [5, 6] and this work. Since we do not assume the route independence, our problem setting is more realistic, making many existing optimization algorithms designed based on the assumption inapplicable to our work.

Comparison items	Existing work [5, 6]	Our work
Market Share Prediction Model	Standard multi-logit model	Deep learning model
Conventional Air Carrier Performance Features	Yes	Yes
Transportation Network Features	No	Yes
Removal of Route Independence Assumption	No	Yes
Optimization Technique	Classical combinatorial optimization techniques	Our proposed adaptive gradient ascent
How to integrate prediction and optimization	Black-box query to prediction model	White-box search

independence assumption is not applicable to our work. Our work does not assume the independence so this work is more realistic.

In the perspective of Knapsack, after excluding the independence assumption, it becomes much more complicated because the value (i.e., market share) of a product (i.e., route) becomes nondeterministic and is influenced by other products (i.e., routes). This makes the current problem more realistic than those studied in the previous work by An et al. However, this change prevents us from applying many existing Knapsack algorithms that have been invented for the simplest case where product values are fixed and independent from each other [8].

One more significant difference is that the optimization algorithm in the related work queries its prediction models whereas both optimization and prediction are integrated on TensorFlow in this new paper. In Table 1, we summarize the differences between the previous work and our work. In addition, Fig. 2 compares their fundamental difference on the algorithm design philosophy. Those existing methods are representative black-box search methods where the query-response strategy is adopted. In this new work, however, the gradients directly flow to update frequencies so its runtime is inherently faster than existing methods.

3 PRELIMINARIES

We introduce our dataset and the state-of-the-art market share prediction model. Our main dataset is the air carrier origin and destination survey (DB1B) dataset released by the US Department of Transportation's Bureau of Transportation Statistics (BTS) [1] and some safety dataset by the National Transportation Safety Board (NTSB) [2]. We refer to Appendix for detailed dataset information.

3.1 Market Share Prediction Model

In this subsection, we describe a popular existing market share prediction model for air transportation markets. Given a route r, the following multinomial logistic regression model is to predict the market share of air carrier C_k in the route:

$$m_{r,k} = \frac{e^{\sum_j w_{r,j} \cdot f_{r,k,j}}}{\sum_i e^{\sum_j w_{r,j} \cdot f_{r,i,j}}} = \frac{exp(\mathbf{w}_r \cdot \mathbf{f}_{r,k})}{\sum_i exp(\mathbf{w}_r \cdot \mathbf{f}_{r,i})}, \tag{1}$$

where $m_{r,k}$ means the market share of air carrier C_k in route r; $f_{r,k,j}$ is the j-th feature of air carrier C_k in route r; and $w_{r,j}$ represents the sensitivity of market share to feature $f_{r,k,j}$ in route r that can be learned from data.

A set of features for air carrier C_k in route r can be represented by a vector $\mathbf{f}_{r,k}$ (see Appendix C for a complete list of $\mathbf{f}_{r,k}$ in our work). We use bold font to denote vectors.

The rationale behind the multi-logit model is that $exp(\mathbf{w}_r \cdot \mathbf{f}_{r,k})$ can be interpreted as passengers' valuation score about air carrier C_k and the market share can be calculated by the normalization of those passengers' valuation scores — this concept is not proposed by us but widely used for the air carrier market share prediction in Business, Operations Research, etc [5, 6, 14, 19, 20].

4 PROPOSED PREDICTION METHOD

We design a neural network-based market share prediction model with transportation network features.

4.1 Air Carrier Transportation Network

There are more than 2,000 routes (e.g., from LAX to JFK) in the US and this creates one large transportation network. Transportation network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a directed graph among airports (i.e., vertices) in $\mathcal{V}$. In particular, we are interested in an air carrier-specific directed transportation network $\mathcal{G}_k$ weighted by its flight frequency values. Thus, $\mathcal{G}_k$ represents the connectivity of air carrier C_k and its edge weight on a certain directional edge means the flight frequency of the air carrier in the route. $\mathcal{G}_k$ can be represented by a weighted adjacency (or frequency) matrix $\mathcal{A}_k$, where each element is a flight frequency from one airport to another.

4.2 Network Features

In this section, we introduce the network features we added to improve the prediction model.

4.2.1 Degree Centrality. As mentioned by earlier works, transportation network connectivity is important in air transportation markets [17, 18]. For instance, the higher the degree centrality of an airport in $\mathcal{G}_k$, the more options the passengers to fly. Thereby, its market share will increase at the routes departing the high degree centrality airport. Therefore, we study how the degree centrality of source and destination airports influences the market share.

Given $\mathcal{A}_k$, the out-degree (resp. in-degree) centrality of i-th airport is the sum of i-th row (resp. column). So this feature calculation can be very easily implemented on Tensorflow or other deep learning platforms.

4.2.2 Ego Network Density. Ego network is very popular for social network analysis [15]. We introduce the concept of ego network first.

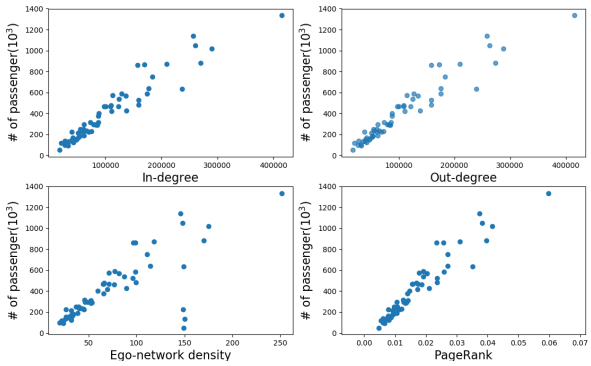


Figure 3: Number of passengers vs. network features. The result summarizes all the airports.

Definition 4.1. Given a vertex v, its *ego network* is an induced subgraph of v and its neighbors. The vertex v is called *ego vertex* (i.e., ego airport in our case). Note that ego networks are also weighted with flight frequency values. The density of an ego network is defined as the sum of edge weights divided by $n(n-1)$ where n is the number of vertices in the ego network.

By the definition, an airport's ego network density is high when the airport and its neighboring airports are well connected all together. It is natural that passengers transit in an airport whose connections are well prepared for their final destinations.

4.2.3 PageRank. PageRank was originally proposed to derive a vertex's importance score based on the random web surfer model [16] — i.e., a web surfer performs a random walk following hyperlinks. We think PageRank is suitable to analyze multi-stop passengers for the following reason.

After normalizing $\mathcal{A}_k$ row-wise, it becomes the transition probability that a random passenger will move following the route. Thus, PageRank is able to capture the importance of an airport.

Fig. 3 depicts the relationships between the network features introduced above and the total number of passengers transported in and out airports by a certain air carrier. We used the DB1B data released by the BTS for the first quarter of 2018 to draw this figure. As shown in Fig. 3, the number of passengers in each airport is highly correlated with the network features (i.e., in-degree, out-degree, ego network density, and PageRank). In conjunction with other classical air carrier performance features, these network features can improve the prediction accuracy by a non-trivial margin.

4.3 Neural Network-based Prediction

Whereas many existing methods rely on classical machine learning approaches, we use the following neural network to predict:

$$\begin{aligned} \mathbf{h}_{r,k}^{(1)} &= \sigma(\mathbf{f}_{r,k}\mathbf{W}^{(0)} + \mathbf{b}^{(0)}), \text{ for initial layer} \\ \mathbf{h}_{r,k}^{(i+1)} &= \mathbf{h}_{r,k}^{(i)} + \sigma(\mathbf{h}_{r,k}^{(i)}\mathbf{W}^{(i)} + \mathbf{b}^{(i)}), \text{ if } i \geq 1 \end{aligned} \tag{2}$$

where σ is ReLU. $\mathbf{W}^{(0)} \in \mathcal{R}^{19\times d}, \mathbf{b}^{(0)} \in \mathcal{R}^{d}, \mathbf{W}^{(i)} \in \mathcal{R}^{d\times d}, \mathbf{b}^{(i)} \in \mathcal{R}^{d}$ are parameters to learn. Note that we use residual connections after the initial layer. For the final activation, we also use the multi-logit regression. From Eq. (1), we replace $\mathbf{f}_{r,k}$ with $\mathbf{h}_{r,k}^{l}$, which denotes the last hidden vector of our proposed neural network, to predict $m_{r,k}$ as follows:

$$m_{r,k} = \frac{exp(\mathbf{w}_r \cdot \mathbf{h}_{r,k}^{l})}{\sum_i exp(\mathbf{w}_r \cdot \mathbf{h}_{r,i}^{l})}, \tag{3}$$

where $\mathbf{w}_r$ is a trainable parameter. We use $\boldsymbol{\theta}_r$ to denote all the parameters of route r in Eqs. (2) and (3).

One thing to mention is that all the network features can be properly calculated on TensorFlow from $\mathcal{A}_k$ before being fed into the neural network. This is the case during the frequency optimization phase which will be described shortly. By changing a frequency in $\mathcal{A}_k$, the entire network feature can be recalculated before the neural network processing as shown in Fig. 1. Therefore, the gradients can directly flow from the prediction models to the frequency matrix through the network feature calculation part. Hereinafter, we use a function $m_{r,k}(\mathcal{A}_k; \boldsymbol{\theta}_r)$ after partially omitting features (such as ticket price, aircraft size, etc.) to denote the predicted market share. Note that the omitted features and $\boldsymbol{\theta}_r$ are considered constant while optimizing frequencies in the next section. We sometimes omit all the inputs and use $m_{r,k}$ for brevity.

5 PROPOSED OPTIMIZATION METHOD

Among many features, the flight frequency is an actionable feature that we are interested in to adjust — see Appendix C for a complete list of features we consider in this work. An actionable feature means a feature that can be freely decided only for one's own purposes. May other features, such as delay time, safety, and so on, cannot be solely decided by an air carrier. Hereinafter, we use $f_{r,k,freq}$ to denote a flight frequency value of air carrier C_k in route r. These frequency values among airports constitute $\mathcal{A}_k$.

5.1 Problem Definition

We solve the following optimization problem to maximize the market influence of air carrier C_k (i.e., the number of passengers transported by C_k) on those routes in $\mathcal{R}$. Given its total budget $budget_k$, we optimize the flight frequency values of the air carrier over multiple routes in $\mathcal{R}$ as follows:

$$\begin{aligned} \max_{f_r^{max} \geq f_{r,k,freq} \geq 0, r\in\mathcal{R}} & \sum_{r\in\mathcal{R}} demand_r \times m_{r,k} \\ \text{subject to } & \sum_{r\in\mathcal{R}} cost_{r,k} \times f_{r,k,freq} \leq budget_k, \end{aligned} \tag{4}$$

where $m_{r,k}$ is the predicted market share of C_k in route r (by our neural network model), $demand_r$ is the number of total passengers in route r from the DB1B dataset, and $cost_{r,k}$ is the unit operational cost of air carrier C_k in route r. f_r^{max} is the maximum flight frequency in route r observed in the DB1B dataset. The adoption of f_r^{max} is our heuristic to prevent overshooting a practically meaningful frequency limit. Note that different air carriers have different unit operational costs in a route r as their efficiency is different and they purchase fuel in different prices — we extract this information from the DB1B dataset.

Eq. (4) shows how we can effectively merge data mining and mathematical optimization. The proposed problem is basically a non-linear optimization and a special case of Integer Knapsack

and resource allocation problems which are all NP-hard [7]. The theoretical complexity of the problem is $O(\prod_{r\in\mathcal{R}} f_r^{max})$, which can be simply written as $O(n^{2,262})$ after assuming $n = f_r^{max}$ in each route for ease of discussion because $|\mathcal{R}| = 2,262$.

Theorem 5.1. *The market influence maximization is NP-hard.*

5.2 Overall Architecture

In Fig. 1, the overall architecture of the proposed optimization idea is shown. The overall workflow is as follows:

(1) Train the market share prediction model in each route, which considers transportation network features.
(2) Fix the prediction models and update the frequency matrix $\mathcal{A}_k$ using the proposed AGA optimizer. We consider other features (such as ticket price, aircraft size, etc.) are fixed while optimizing frequencies.

The adoption of network features makes many classical combinatorial optimization techniques inapplicable to our work because the route independent assumption does not hold any more. Even worse, our objective function consists of highly non-linear neural networks. Therefore, our problem becomes a challenging non-linear optimization problem. We shortly describe how to solve such a large-scale and difficult optimization problem.

5.3 Gradient-based Optimization

We solve the problem in Eq. (4) on a deep learning platform using our AGA method in Algorithm (1). But one problem in this approach is how to consider the budget constraint. We design two workarounds based on i) Lagrangian function (LF) and ii) rectified linear unit (ReLU).

In our heuristic, we covert integer frequency variables to real variables and use the `clip_by_value` function of TensorFlow to restrict the frequency in r into $[0, f_r^{max}]$ during the optimization process. As the optimized frequencies by our method will be real numbers, *we round down to convert them to integers* and not to violate the budget limit at the end of the optimization process i.e., a continuous relaxation from integer frequencies. We now describe how to solve the continuous-relaxed problem.

5.3.1 Lagrangian Function (LF)-based Heuristic: The method of Lagrange multiplier is a popular method to maximize concave functions (or some special non-concave functions) with constraints [9, 10]. However, we cannot apply the method to our work because our objective function consists of highly non-linear neural networks. Therefore, we adopt only the Lagrangian function from the method and develop our own heuristic search method. The following Lagrangian function can be defined in our case:

$$L = o(\mathcal{A}_k) - \lambda c(\mathcal{A}_k), \tag{5}$$

where λ is called a Lagrange multiplier, and

$$\begin{aligned} o(\mathcal{A}_k) &= \sum_{r\in\mathcal{R}} demand_r \times m_{r,k}, \\ c(\mathcal{A}_k) &= \sum_{r\in\mathcal{R}} \left(cost_{r,k} \times f_{r,k,freq}\right) - budget_k. \end{aligned} \tag{6}$$

Basically, the Lagrange multiplier λ can be systematically decided, if the objective function $o(\mathcal{A}_k)$ is in simple forms, and we can find the optimal solution of the original constrained problem. However, this is not the case in our work due to the complicated nature of neural networks and the objective function from them, and our goal is to solve the optimization problem on TensorFlow for the purpose of increasing scalability, aided by our scalable AGA optimization technique. Thus, we propose the following regularized problem and develop a heuristic search method:

$$\max_{f_r^{max}\geq f_{r,k,freq}\geq 0, r\in\mathcal{R}} \quad \min_{\lambda} \quad L + \delta\lambda^2 \tag{7}$$

where $\delta \geq 0$ is a weight for the regularization term. Note that our definition is different from the original Lagrangian function. The inner minimization part has been added by us to prevent that λ becomes too large. One way to solve Eq. (7) is to alternately optimize flight frequencies (i.e., the outer maximization) and λ (i.e., the inner minimization), which implies that Eq. (7) be basically a two-player max-min game. We further improve Eq. (7) and derive a simpler but equivalent formulation that does not require the alternating maximization and minimization shortly in Eq. (9).

Theorem 5.2. *Let $\mathcal{A}_k$ be a matrix of flight frequencies. The optimal solution of the max-min problem in Eq. (7) is the same as the optimal solution of the following problem:*

$$\max_{f_r^{max}\geq f_{r,k,freq}\geq 0, r\in\mathcal{R}} \quad o(\mathcal{A}_k) - \frac{c(\mathcal{A}_k)^2}{4\delta}. \tag{8}$$

For simplicity, let $\beta = \frac{1}{2\delta}$ and we can rewrite Eq. (8) as follows:

$$\max_{f_r^{max}\geq f_{r,k,freq}\geq 0, r\in\mathcal{R}} \quad \bar{L}_{Lagrange}, \tag{9}$$

where $\bar{L}_{Lagrange} = o(\mathcal{A}_k) - \beta\frac{c(\mathcal{A}_k)^2}{2}$.

Note that maximizing Eq. (9) is equivalent to solving the max-min problem in Eq. (7) so we implement only Eq. (9) and optimize it using the proposed AGA method that will be described in the next subsection.

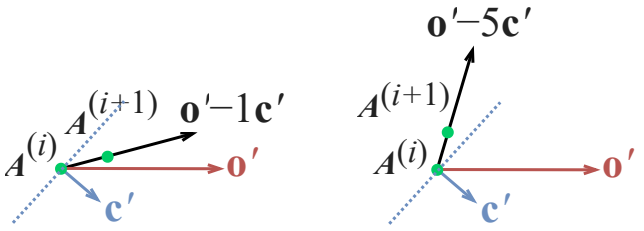
5.3.2 Rectified Linear Unit (ReLU)-based Heuristic: ReLU is used to rectify an input value by taking its positive part for neural networks. This property can be used to impose a penalty if the budget limit constraint is violated as follows:

$$\max_{f_r^{max}\geq f_{r,k,freq}\geq 0, r\in\mathcal{R}} \quad \bar{L}_{ReLU}, \tag{10}$$

where $\bar{L}_{ReLU} = o(\mathcal{A}_k) - \beta R(c(\mathcal{A}_k))$ and $R(\cdot)$ is the rectified linear unit.

5.4 β Selection and Adaptive Gradient Ascent

We propose the AGA method, which basically uses the gradients of $\bar{L}_{Lagrange}$ or $\bar{L}_{ReLU}$ w.r.t. flight frequencies to optimize them. In both methods, the coefficient β needs to be *dynamically* adjusted to ensure the budget limit rather than being fixed to a constant. For example, one gradient ascent update will increase flight frequencies even after a cost overrun if β is not large enough. Whenever there is any cost overrun, β should be set to such a large enough value that the total cost is decreased.



(a) $\beta = 1$ is not enough to decrease cost (b) $\beta = 5$ is enough to decrease cost

Figure 4: Suppose that there is a small cost overrun with $\mathcal{A}^{(i)}$, which denotes a frequency matrix at i-th gradient ascent iteration. The norm of $\mathbf{c}'$ is smaller than that of $\mathbf{o}'$ and the gradient ascent update cannot remove the cost overrun if β is small (e.g., $\beta = 1$ in (a)). However, if β is large enough (e.g., $\beta = 5$ in (b)), the gradient ascent update can reduce the cost overrun. Note that $\mathcal{A}^{(i+1)}$ is located behind $\mathcal{A}^{(i)}$ w.r.t. the blue dotted line perpendicular to $\mathbf{c}'$ in (b), which means a reduced cost overrun. We dynamically adjust β to decrease the cost, if any cost overrun, while sacrificing the objective as little as possible.

For the sake of our convenience, we will use $\mathbf{o}'$ and $\mathbf{c}'$ to denote the gradients of objective and cost overrun penalty term as follows:

$$\mathbf{o}' = \nabla o(\mathcal{A}_k),$$

$$\mathbf{c}' = \begin{cases} \nabla \frac{c(\mathcal{A}_k)^2}{2}, & \text{if the Lagrangian function-based method} \\ \nabla R(c(\mathcal{A}_k)), & \text{if the ReLU-based method.} \end{cases}$$

Fig. 4 shows an illustration of why we need to adjust β. As shown, if the directions of the two gradients $\mathbf{c}'$ and $\mathbf{o}' - \beta\mathbf{c}'$, where $\beta = 5$, are opposite, the cost overrun will decrease after one gradient ascent update. If β is too small, the cost overrun does not decrease in the example.

We also do not distinguish between $\bar{L}_{Lagrange}$ and $\bar{L}_{ReLU}$ in this section because the algorithm proposed in this section is commonly applicable to both the Lagrangian function and ReLU-based methods. We denote them simply as $\bar{L}$ in this section.

The gradients of $\bar{L}$ w.r.t. $\mathcal{A}_k$ are made of two components $\mathbf{o}'$ and $-\beta\mathbf{c}'$, where $\mathbf{o}'$ increases the market influence and $-\beta\mathbf{c}'$ reduces the cost overrun. Typically, the market influence increases as the frequencies $\mathcal{A}_k$ increase. So β needs to be properly selected such that the frequencies are updated (by the proposed AGA method) to reduce the cost once the total cost exceeds the budget during the gradient-based update process. This requires that the overall gradients $\mathbf{o}' - \beta\mathbf{c}'$ suppresses an increase in $c(\mathcal{A}_k)$. More precisely, it requires that the directional derivative of $c(\mathcal{A}_k)$ along the vector $\mathbf{o}' - \beta\mathbf{c}'$ (or the dot product of $\mathbf{o}' - \beta\mathbf{c}'$ and $\mathbf{c}'$) is negative — if two vectors have different directions, their dot product is negative.

Therefore, we want $\mathbf{c}' \cdot (\mathbf{o}' - \beta\mathbf{c}') < 0$. From it, we can rewrite the inequality w.r.t. β and we have

$$\beta > \frac{\mathbf{c}' \cdot \mathbf{o}'}{\mathbf{c}' \cdot \mathbf{c}'}. \tag{11}$$

Algorithm 1: Adaptive gradient ascent (AGA)

Input: γ
Output: $\mathcal{A}_k$

```
1  Initialize A_k;                          /* Initialize freqs */
2  β ← 0;                                   /* Initialize β */
3  while until convergence do
4  |   A_k ← A_k + γ∇L̄;                    /* Gradient ascent */
5  |   if c(A_k) > 0 then
6  |   |   β ←Eq. (13)
7  |   else
8  |   |   β ← 0;
9  |   end
10 end
```

Note that Eq. (11) does not include the equality condition but requires that β is strictly larger than its right-hand side. To this end, we introduce a positive value $\epsilon > 0$ as follows:

$$\beta = \frac{\mathbf{o}' \cdot \mathbf{c}'}{\mathbf{c}' \cdot \mathbf{c}'} + \epsilon, \tag{12}$$

where ϵ is a positive hyper-parameter in our method.

On the other hand, we need to ensure that β is getting closer to zero when the algorithm is approaching an optimal solution of $\mathcal{A}_k$. To do this, we further modify it as follows:

$$\beta = \frac{\mathbf{o}' \cdot \mathbf{c}'}{\mathbf{c}' \cdot \mathbf{c}'} + c(\mathcal{A}_k)\epsilon. \tag{13}$$

Note that $c(\mathcal{A}_k)\epsilon$ becomes a very trivial value if $c(\mathcal{A}_k)$ is very small. This specific setting prevents the situation that an ill-chosen large ϵ decreases flight frequencies too much given a very small cost overrun $c(\mathcal{A}_k) \approx 0$.

The proposed AGA method is presented in Algorithm 1. The optimization of frequencies occurs at line 4 and other lines are for dynamically adjusting β. We take a solution around 500 epochs when the cost overrun is not positive. 500 epochs are enough to reach a solution point in our experiments.

Theorem 5.3. *Algorithm 1 is able to find a feasible solution of the original problem in Eq.* (4).

6 EXPERIMENTS

In this section, we introduce experimental environments and results for both the prediction and the optimization. We collected our data for 10 years from the website [1]. We predict the market share and optimize the flight frequency in the last month of the dataset after training with all other month data.

In our dataset, there are 2,262 routes and more than 10 air carriers. We predict and optimize for the top-4 air carriers among them considering their influences on the US domestic air markets. We ignore other regional/commuter level air carriers.

Our detailed software and hardware environments are as follows: Ubuntu 18.04.1 LTS, Python ver. 3.6.6, Numpy ver. 1.14.5, Scipy ver. 1.1.0, Pandas ver. 0.23.4, Matplotlib ver.3.0.0, Tensorflow-gpu ver. 1.11.0, CUDA ver. 10.0, NVIDIA Driver ver. 417.22. Three machines with i9 CPU and GTX1080Ti are used.

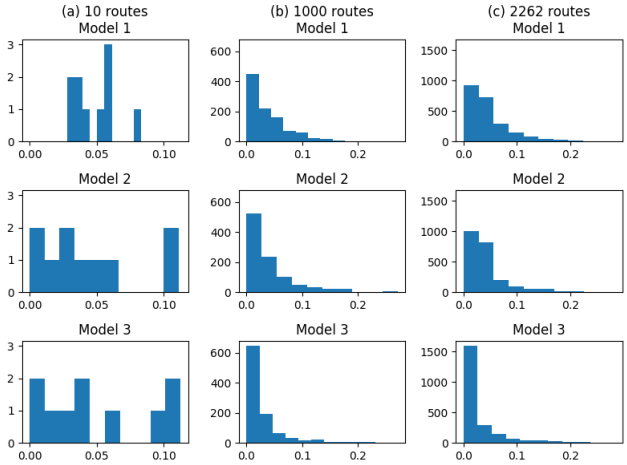


Figure 5: Histogram of RMSE scores — lower values are preferred. X-axis is the RMSE score and Y-axis is the number of routes.

Table 2: Median/Average RMSE and R^2. The up-arrow (resp. down-arrow) means higher (resp. lower) is better. The best results are indicated in bold font.

		Median RMSE ↓	R^2 ↑	Mean RMSE ↓
10 routes	TheilSen	0.048	0.944	0.052
	AdaBoost	0.029	0.970	0.027
	RandomForest	0.029	**0.979**	0.025
	Model1 [19]	0.026	0.965	**0.024**
	Model2 [20]	0.035	0.953	0.030
	Model3 (Ours)	**0.023**	0.899	0.026
	Model3 (No Net.)	0.026	0.884	0.027
1,000 routes	TheilSen	0.080	0.855	0.087
	AdaBoost	0.021	0.964	0.033
	RandomForest	0.024	0.968	0.033
	Model1 [19]	0.021	0.957	0.033
	Model2 [20]	0.020	0.983	0.035
	Model3 (Ours)	**0.010**	**0.988**	**0.025**
	Model3 (No Net.)	0.019	0.978	0.030
2,262 routes	TheilSen	0.0813	0.707	0.088
	AdaBoost	0.017	0.933	**0.031**
	RandomForest	0.014	0.942	**0.031**
	Model1 [19]	0.033	0.944	0.041
	Model2 [20]	0.030	0.976	0.033
	Model3 (Ours)	**0.007**	**0.983**	0.038
	Model3 (No Net.)	0.013	0.969	0.040

6.1 Market Share Prediction

6.1.1 Baseline Methods. We compare our proposed model with two baseline prediction models. Model1 [19] considers air carrier's frequency, delay, and safety. Model2 [20] studies the effect of aircraft size and seat availability on market share and considers all other variables such as price and frequency. Model1 and Model2 are conventional methods based on multi-logit regression and they are trained using numerical solvers. Model3 is a neural network-based model created by us and uses the network features as well.

To train the market share prediction models, we use the learning rate of 1e-4 which decays with a ratio of 0.96 every 100 epochs. The number of layer in our neural network is $l = \{3, 4, 5\}$ and the dimensionality of the hidden vector is $d = \{16, 32\}$. We train 1,000 epochs for each model and use the Xavier initializer [13] for initializing weights and the Adam optimizer for updating weights. We used the cross validation method to choose the best one, which means given a training set with N months, we choose a random month and validate with the selected month after training with all other $N - 1$ months. We repeat this N times.

In addition, we test other standard regression algorithms as well. In particular, we are interested in testing some robust regression algorithms such as TheilSen, AdaBoost Regression, and RandomForest Regression. We also use the same cross validation method.

6.1.2 Experimental Results. Fig. 5 shows the histogram of RMSE scores for Model1, 2, and 3. We experimented three scenarios (i.e., top-10 routes, top-1,000 routes, and top-2,262 routes in terms of the number of passengers). Our Model3 shows a higher density in low-RMSE regions than other models.

The median/average root-mean-square error (RMSE) and R^2 scores are summarized in Table 2. Our Model3 has much better median RMSE and R^2 scores than other models (especially for the largest scale prediction with 2,262 routes). Sometimes our mean RMSE is worse than other baselines. However, we think this is not significant because our low median RMSE says that it is better than others in the majority of routes. In particular, we show the median RMSE of 0.007 for the 2,262-route predictions vs. 0.030 by Model2. RandomForest also shows reasonable accuracy in many cases.

For the top-10 routes, most models have good performance. This is because it is not easy for our model to have reliable network features only with the 10 routes. However, our main goal is to predict accurately in a larger scale prediction.

We also compare the accuracy of our proposed model without the network features, denoted with "No Net." in the table. When we do not use any network features, the accuracy of market share predictions slightly decreases. Considering the scale of the market size, however, a few percentage errors can result in a big loss in the optimization phase. Therefore, our proposed prediction model is the most suitable to be used to define the objective function of our proposed optimization problem.

6.2 Market Influence Maximization

6.2.1 Baseline Methods. Dynamic programming, branch and bound, and GroupGreedy were used to solve a similar problem in [5, 6]. However, all these algorithms assume that routes are independent, which is not the case in our work because we use the network features. Therefore, their methods are not applicable to our work (see Section 2.2).

Therefore, we describe two baseline methods: greedy and an exhaustive algorithm. Greedy methods are effective in many optimization problems. In particular, greedy provides an approximation ratio of around 63% for submodular minimization. Unfortunately, our optimization is not a submodular case. Due to its simplicity, however, we compare with the following greedy method, which iteratively chooses a route with the maximum marginal increment of market influence and increases its flight frequency by α. In general, the step size α is 1. For faster convergence, however, we test various $\alpha = \{1, 10\}$. The complexity of the greedy algorithm is $O(\frac{budget_k \cdot N_k}{\alpha \cdot avg\_cost_k})$, where N_k is the number of routes and avg_cost_k

Table 3: Optimization results for the top-3 routes. Multiplying by 10 will lead to the real scale of passenger numbers because the DB1B database includes 10% random samples of air tickets. LF and ReLU mean our Lagrangian function and ReLU-based methods, respectively.

		Carrier 1	Carrier 2	Carrier 3	Carrier 4
# of Passengers	Ground Truth	4,960	307	1,792	3,124
	LF, Real_Init (Ours)	4,964	308	1,842	3,126
	ReLU, Real_Init (Ours)	4,961	**310**	1,854	**3,144**
	LF, Zero_Init (Ours)	4,970	308	**1,891**	3,139
	ReLU, Zero_Init (Ours)	4,961	**310**	**1,891**	**3,144**
	Greedy, Zero_Init, $\alpha = 1$	4,967	**310**	**1,891**	**3,144**
	Greedy, Zero_Init, $\alpha = 10$	**4,972**	**310**	**1,891**	**3,144**
	Brute-force, Zero_Init, $\alpha = 5$	**4,972**	N/A	N/A	N/A
	Brute-force, Zero_Init, $\alpha = 10$	**4,972**	**310**	**1,891**	**3,144**

Table 4: Optimized number of passengers for the top-10 routes.

		Carrier 1	Carrier 2	Carrier 3	Carrier 4
# of Passengers	Ground Truth	16,924	4,022	20,064	29,419
	LF, Real_Init (Ours)	18,612	4,054	20,552	30,220
	ReLU, Real_Init (Ours)	18,618	**5,024**	**20,703**	**30,269**
	LF, Zero_Init (Ours)	18,583	4,259	20,549	30,074
	ReLU, Zero_Init (Ours)	**18,643**	**5,024**	20,323	**30,269**
	Greedy, Zero_Init, $\alpha = 1$	17,016	**5,024**	20,515	29,519
	Greedy, Zero_Init, $\alpha = 10$	18,078	**5,024**	20,515	**30,269**

Table 5: Running time (in sec.) for the top-10 routes.

	Carrier 1	Carrier 2	Carrier 3	Carrier 4
LF, Real_Init (Ours)	40.77	42.52	41.86	41.70
ReLU, Real_Init (Ours)	**40.75**	41.48	**40.67**	44.30
LF, Zero_Init (Ours)	43.10	42.45	40.94	40.37
ReLU, Zero_Init (Ours)	40.98	**39.90**	40.49	**40.31**
Greedy, Zero_Init, $\alpha = 1$	910.12	191.12	1,074.95	1,001.04
Greedy, Zero_Init, $\alpha = 10$	89.47	20.14	107.82	101.01

is the average cost for air carrier k over the routes. However, this greedy is still a black-box method, whose efficiency is worse than our white-box method.

We can also use a brute-force algorithm when the number of routes is small. Given three routes $\{r_1, r_2, r_3\}$, for instance, the possible number of solutions is $f_{r_1}^{max} \times f_{r_2}^{max} \times f_{r_3}^{max}$. It is already a very large search space because each $f_{r_i}^{max}$ is several hundreds for a popular route in a month. However, we do not need to test solutions one by one. We create a large tensor of $|\mathcal{R}| \times |\mathcal{R}| \times q$ dimensions, where q is the number of queries, and query q solutions at the same time. In general, GPUs can solve the large query quickly. Even with GPUs, however, we cannot query more than a few routes because the search space volume exponentially grows. We also use the step size $\alpha = \{5, 10\}$. $\alpha = 1$ is not feasible in the brute-force search even with state-of-the-art GPUs. Thus, the complexity becomes $O(\frac{f_{r_1}^{max}}{\alpha} \times \frac{f_{r_2}^{max}}{\alpha} \times \frac{f_{r_3}^{max}}{\alpha})$.

6.2.2 Hyperparameter Setup. For all methods, we let the flight frequency $f_{r,k,freq}$ of air carrier C_k in route r on or below the maximum frequency f_r^{max} observed in the DB1B database. This is very important to ensure feasible frequency values because too high frequency values may not be accepted in practice due to limited capacity of airports. This restriction can be implemented using the $clip\_by\_value(\cdot)$ function of Tensorflow.

In addition, we need to properly initialize frequency values in Algorithm 1. We test two ways to initialize frequencies: i) Real_Init initializes the flight frequency values with the ground-truth values observed in the dataset, and ii) Zero_Init initializes all the frequencies to zeros. In all methods, we set the total budget to the ground-truth budget.

We tested $\epsilon = \{1, 100, 1000\}$ but there is no significant difference on the achieved final optimization values. For the following experiments, we choose $\epsilon = 1000$ to speed up the optimization process. We use 10 for the learning rate γ and run 500 epochs.

One more thing is that the DB1B database includes 10% random samples of air tickets[2] so our reported passenger numbers multiplied by 10 will be the real scale. In this paper, we list values in the original scale of the DB1D database for better reproducibility.

6.2.3 Experimental Results. We first compare all the aforementioned methods in a small sized problem with only 3 routes. Especially, the brute-force search is possible only for this small problem.

For the top-3 route optimization, we choose the top-3 biggest routes and the top-4 air carriers in terms of the number of passengers transported and optimize the flight frequencies in the 3 routes for each air carrier for the last month of our dataset. In Table 3, detailed optimized market influence values are listed for various methods. Surprisingly, all methods mark similar values. We think all methods are good at solving this small size problem. However, the brute-force method is not feasible for some cases where the maximum frequency limits f_r^{max} in the routes are large — we mark with 'N/A' for those whose runtime is prohibitively large.

Experimental results of the top-10 route optimization are summarized in Table 4. Our method based on the ReLU activation produces the best results for all the top-4 air carriers. Our Lagrangian function (LF)-based optimization also produces many reasonable results better than Greedy. Greedy shows the worst performance in this experiment. In Table 5, their runtimes are also reported. Our method is 2-22 times faster than the Greedy except Carrier 2 with $\alpha = 10$.

For the top-1,000 and 2,262 routes, experimental results are listed in Tables 6 and 7. Our methods produce the best optimized value in the least amount of time. In particular, our method is about 690 times faster than the Greedy with $\alpha = 10$ at Carrier 4. Greedy is not feasible for 2,262 routes.

Our method shows a *(sub-)linear* increment of runtime w.r.t. the number of routes. It takes about 40 seconds for the top-10 routes and 400 seconds for the top-1,000 routes. When the problem size becomes two orders of magnitude larger from 10 to 1,000, the runtime increases only by one order of magnitude. Considering that we solve a NP-hard problem, the sub-linear runtime increment is an outstanding achievement. Moreover, our method consistently shows the best optimized values in almost all cases.

Greedy is slower than our method due to its high complexity $O(\frac{budget_k \cdot N_k}{\alpha \cdot avg\_cost_k})$ as described in Sec. 6.2.1. When the budget limit $budget_k$ and the number of routes N_k are large, it should query the prediction models many times, which significantly delays its

[2]See the overview section in https://www.transtats.bts.gov/DatabaseInfo.asp?DB_ID=125.

Table 6: Optimized number of passengers for the top-1,000 and 2,262 routes. Greedy with $\alpha = 1$ is not feasible in this scale of experiments.

	Carrier 1		Carrier 2		Carrier 3		Carrier 4	
	1000 routes	2262 routes	1000 routes	2262 routes	1000 routes	2262 routes	1000 routes	2262 routes
LF, Real_Init (Ours)	429,581	487,475	**225,623**	307,815	388,864	447,633	546,742	723,522
ReLU, Real_Init (Ours)	431,261	489,684	**225,623**	**307,881**	**390,239**	**448,421**	547,623	725,526
LF, Random_Init (Ours)	426,683	**498,511**	225,057	306,268	373,756	435,277	548,310	**726,142**
ReLU, Random_Init (Ours)	**434,154**	492,784	224,603	306,963	380,318	447,656	**549,092**	721,100
Greedy, Zero_Init, $\alpha = 10$	428,196	N/A	225,322	N/A	385,607	N/A	516,348	N/A

Table 7: Running time (in sec.) for the top-1,000 and 2,262 routes scenarios.

	Carrier 1		Carrier 2		Carrier 3		Carrier 4	
	1000 routes	2262 routes	1000 routes	2262 routes	1000 routes	2262 routes	1000 routes	2262 routes
LF, Real_Init (Ours)	440.21	875.15	450.30	964.00	**451.10**	951.56	439.56	940.17
ReLU, Real_Init (Ours)	439.18	908.66	452.39	947.35	453.50	**937.25**	438.75	950.60
LF, Random_Init (Ours)	**438.06**	**878.73**	451.15	947.35	453.39	949.80	440.17	948.20
ReLU, Random_Init (Ours)	442.12	891.05	**448.85**	**936.29**	452.47	967.23	**438.49**	**928.13**
Greedy, Zero_Init, $\alpha = 10$	84,643.31	N/A	13,414.46	N/A	35,116.56	N/A	302,272.43	N/A

solution search time. Therefore, Greedy is a classical black-box search method whose efficiency is much worse than our proposed method. One can consider our method as a white-box search method because the gradients flow directly to update flight frequencies.

7 CONCLUSION

We presented a prediction-driven optimization framework for maximizing air carriers' market influence, which includes neural network-based market share prediction models by adding transportation network features and innovates large-scale optimization techniques through the proposed AGA method. Our approach suggests a way to unify data mining and mathematical optimization. Our network feature-based prediction shows better accuracy than existing methods. Our AGA method can optimize for all the US domestic routes in our dataset at the same time whereas state-of-the-art methods are applicable to at most tens of routes.

ACKNOWLEDGMENTS

Noseong Park is the corresponding author. This work of Jinsung Jeon, Seoyoung Hong, and Noseong Park was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2020-0-01361, Artificial Intelligence Graduate School Program (Yonsei University)). The work of Thai Le and Dongwon Lee was in part supported by NSF awards #1909702, #1940076, #1934782, and #2114824.

REFERENCES

[1] [n.d.]. Bureau of Transportation Statistics. ([n. d.]). http://www.rita.dot.gov/bts/data_and_statistics/by_mode/airline_and_airports/index.html.
[2] [n.d.]. National Transportation Safety Board. ([n. d.]). http://www.ntsb.gov/_layouts/ntsb.aviation/index.aspx.
[3] J. A. Abdella, N. Zaki, and K. Shuaib. 2018. Automatic Detection of Airline Ticket Price and Demand: A review. In *2018 International Conference on Innovations in Information Technology (IIT)*. 169–174.
[4] H. Amin, K. M. Curtis, and B. R. Hayes-Gill. 1997. Piecewise linear approximation applied to nonlinear function of a neural network. *IEE Proceedings - Circuits, Devices and Systems* 144, 6 (Dec 1997), 313–317.
[5] Bo An, Haipeng Chen, Noseong Park, and V.S. Subrahmanian. 2016. MAP: Frequency-Based Maximization of Airline Profits Based on an Ensemble Forecasting Approach. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
[6] Bo An, Haipeng Chen, Noseong Park, and V. S. Subrahmanian. 2017. Data-Driven Frequency-Based Airline Profit Maximization. *ACM Trans. Intell. Syst. Technol.* 8, 4 (March 2017), 61:1–61:28.
[7] Sanjeev Arora and Boaz Barak. 2009. *Computational Complexity: A Modern Approach* (1st ed.). Cambridge University Press, New York, NY, USA.
[8] Kyriakos Axiotis and Christos Tzamos. 2019. Capacitated Dynamic Programming: Faster Knapsack and Graph Algorithms. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 132)*, Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi (Eds.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 19:1–19:13.
[9] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. 2011. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Found. Trends Mach. Learn.* 3, 1 (Jan. 2011), 1âĂŞ122. https://doi.org/10.1561/2200000016
[10] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press, USA.
[11] F. Ciliberto, C. Murry, E. Tamer, and Centre for Economic Policy Research (Great Britain). 2018. *Market Structure and Competition in Airline Markets*. Centre for Economic Policy Research.
[12] Sougata Deb. 2017. Analytical Ideas to Improve Daily Demand Forecasts: A Case Study. In *Intelligent Information and Database Systems*, Ngoc Thanh Nguyen, Satoshi Tojo, Le Minh Nguyen, and Bogdan Trawiński (Eds.). Springer International Publishing, Cham, 23–32.
[13] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATSâĂŹ10). Society for Artificial Intelligence and Statistics*.
[14] Mark Hansen. 1990. Airline competition in a hub-dominated environment: An application of noncooperative game theory. *Transportation Research Part B: Methodological* 24, 1 (1990), 27–43.
[15] Jure Leskovec and Julian J. Mcauley. 2012. Learning to Discover Social Circles in Ego Networks. In *Advances in Neural Information Processing Systems 25*.
[16] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank Citation Ranking: Bringing Order to the Web.
[17] Aura Reggiani, Peter Nijkamp, and Alessandro Cento. 2010. Connectivity and concentration in airline networks: a complexity analysis of Lufthansa's network. *European Journal of Information Systems* 19, 4 (2010), 449–461.
[18] Manasi Sapre and Nita Parekh. 2011. Analysis of Centrality Measures of Airport Network of India. In *Pattern Recognition and Machine Intelligence*, Sergei O. Kuznetsov, Deba P. Mandal, Malay K. Kundu, and Sankar K. Pal (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 376–381.
[19] Yoshinori Suzuki. 2000. The relationship between on-time performance and airline market share: a new approach. *Transportation Research Part E: Logistics and Transportation Review* 36, 2 (2000), 139–154.
[20] Wenbin Wei and Mark Hansen. 2005. Impact of aircraft size and seat availability on airlines demand and market share in duopoly markets. *Transportation Research Part E: Logistics and Transportation Review* 41, 4 (2005), 315–327.

A PROOFS

Theorem A.1. *The proposed market influence maximization is NP-hard.*

Proof. We prove the theorem by showing that an arbitrary Integer Knapsack problem instance can be reduced to a special case of our market influence maximization problem.

In an Integer Knapsack problem, there are n product types and each product type p has a value v_p and a cost c_p. In particular, there exist an enough number of product instances for a product type so we can choose multiple instances for a certain product type. Given a budget B, we can choose as many instances as we want such that the sum of the product values are maximized.

This problem instance can be reduced to a market influence maximization by letting a product type p be a route r, c_p be $cost_{r,k}$, and v_p be a deterministic increment of market influence by increasing the frequency by one.

Therefore, the proposed market influence maximization problem is NP-hard. □

Theorem A.2. *Let $\mathcal{A}_k$ be a matrix of flight frequencies. The proposed max-min method in Eq.* (7) *is equivalent to*

$$\max_{f_r^{max} \geq f_{r,k,freq} \geq 0, r \in \mathcal{R}} o(\mathcal{A}_k) - \frac{c(\mathcal{A}_k)^2}{4\delta}.$$

Proof. First we rewrite Eq. (7) as follows:

$$\max_{f_r^{max} \geq f_{r,k,freq} \geq 0, r \in \mathcal{R}} \min_{\lambda} \quad o(\mathcal{A}_k) - \lambda c(\mathcal{A}_k) + \delta\lambda^2. \tag{14}$$

Let us fix $\mathcal{A}_k$ then Eq. (14) becomes a quadratic function (parabola) w.r.t. λ. It is already known that the optimal solution to minimize the quadratic function given a fixed $\mathcal{A}_k$ is achieved when its derivative w.r.t. λ is zero, i.e., $\nabla_\lambda o(\mathcal{A}_k) - \lambda c(\mathcal{A}_k) + \delta\lambda^2 = -c(\mathcal{A}_k) + 2\delta\lambda = 0$. Therefore, the optimal form of λ can be derived as $\hat{\lambda} = \frac{c(\mathcal{A}_k)}{2\delta}$.

Let us substitute λ for its optimal form $\hat{\lambda}$ in Eq. (14) and the inner minimization will disappear as follows:

$$\max_{f_r^{max} \geq f_{r,k,freq} \geq 0, r \in \mathcal{R}} \quad o(\mathcal{A}_k) - \frac{c(\mathcal{A}_k)^2}{4\delta} \tag{15}$$

□

Theorem A.3. *Algorithm 1 is able to find a feasible solution of the original problem in Eq.* (4).

Proof. In Eq. (11), we choose a β configuration that meets $\mathbf{c}' \cdot (\mathbf{o}' - \beta\mathbf{c}') < 0$. The frequency matrix $\mathcal{A}_k$ is updated by the gradient ascent rule, denoted $\mathcal{A}_k = \mathcal{A}_k + \gamma(\mathbf{o}' - \beta\mathbf{c}')$. However, the directions of $\mathbf{o}' - \beta\mathbf{c}'$ and $\mathbf{c}'$ are opposite to each other (because their dot-product is negative), which means the gradient ascent update will decrease the cost overrun term $c(\mathcal{A}_k)$ as illustrated in Fig. 4.

Therefore, after applying the proposed gradient ascent multiple times any cost overrun can be removed. Our algorithm stops at the first solution whose cost overrun is not positive after at least 500 epochs. Therefore, Algorithm 1 is able to find a feasible solution that meets the budget constraint and its termination is guaranteed. □

B DATASETS

Our main dataset is the air carrier origin and destination survey (DB1B) dataset released by the US Department of Transportation's Bureau of Transportation Statistics (BTS) [1]. They release 10% of tickets sold in the US every quarter of year for research purposes, in conjunction with much detailed air carrier information. Itemized operational expenses of air carrier are very well summarized in the dataset and for instance, we can know that how much each air carrier had paid for fuel and attendants and what kinds of air crafts were used by a certain air carrier in a certain route. Air carrier's performance is also one important type of information in the dataset. We also use some safety dataset by the National Transportation Safety Board (NTSB) [2]. We list the links to the web pages where we downloaded our dataset.

(1) https://www.transtats.bts.gov/Tables.asp?DB_ID=125
   (a) DB1B is one of the main tables in the database and contains randomly sampled itineraries.
(2) https://www.transtats.bts.gov/Tables.asp?DB_ID=110
   (a) T-100 Domestic Market contains detailed information about markets (i.e., routes or segments).
(3) https://www.transtats.bts.gov/Tables.asp?DB_ID=120
   (a) Airline On-Time Performance Data contains detailed delay and cancel information about certain flights.
(4) https://www.transtats.bts.gov/Tables.asp?DB_ID=135
   (a) Air Carrier Financial Reports data contains the operational expense for most U.S. air carriers.
(5) https://www.ntsb.gov/_layouts/ntsb.aviation/index.aspx
   (a) The NTSB aviation accident database contains all civil aviation accident records ever since 1962.

B.1 Data Crawling

The Bureau of Transportation Statistics (BTS) collect all the domestic air tickets sold in the US and some additional management information and release the following three main tables: Coupon, Market, and Ticket. The Coupon table, which contains 880,384,622 rows in total, provides coupon-specific information for each domestic itinerary of the Origin and Destination Survey, such as the operating carrier, origin and destination airports, number of passengers, fare class, coupon type, trip break indicator, and distance. The Market table, which has 535,639,256 rows, contains directional market characteristics of each domestic itinerary of the Origin and Destination Survey, such as the reporting carrier, origin and destination airport, prorated market fare, number of market coupons, market miles flown, and carrier change indicators, and the Ticket table, which has 303,276,607 rows, contains summary characteristics of each domestic itinerary on the Origin and Destination Survey, including the reporting carrier, itinerary fare, number of passengers, originating airport, roundtrip indicator, and miles flown. Those thee tables share a set of common columns, i.e., primary-foreign key relationships in a database, and thus can be merged into one large table. Sometime airline names are changed so we use the unique identifiers assigned by the US governments rather than their names.

C FINAL FEATURE SET IN OUR PREDICTION

The complete elements of $\mathbf{f}_{r,k}$ we use for our prediction are as follows so $\mathbf{f}_{r,k}$ is a 19-dimensional vector:

(1) $f_{r,k,0}$: Average ticket price
(2) $f_{r,k,1}$: Flight frequency
(3) $f_{r,k,2}$: Delay ratio
(4) $f_{r,k,3}$: Average delayed time in minutes
(5) $f_{r,k,4}$: Flight cancel ratio
(6) $f_{r,k,5}$: Flight divert ratio
(7) $f_{r,k,6}$: Total number of fatal cases
(8) $f_{r,k,7}$: Total number of serious accident cases
(9) $f_{r,k,8}$: Total number of minor accident cases
(10) $f_{r,k,9}$: Average aircraft size in terms of number of seats per flight
(11) $f_{r,k,10}$: Average seat availability percentage which is not occupied by connecting passengers
(12) $f_{r,k,11}$: In-degree of the source airport
(13) $f_{r,k,12}$: In-degree of the destination airport
(14) $f_{r,k,13}$: Out-degree of the source airport
(15) $f_{r,k,14}$: Out-degree of the destination airport
(16) $f_{r,k,15}$: PageRank of the source airport
(17) $f_{r,k,16}$: PageRank of the destination airport
(18) $f_{r,k,17}$: Ego network density of the source airport
(19) $f_{r,k,18}$: Ego network density of the destination airport

In our work, we optimize $f_{r,k,1}$ in each route to maximize the sum of market shares.

D EXPERIMENTAL ENVIRONMENTS

We introduce detailed environments we conducted our experiments on. We first describe software and hardware environments and then list detailed hyper-parameters.

Our detailed software environments are as follows:

(1) Ubuntu 18.04.1 LTS
(2) Python ver. 3.6.6
(3) Numpy ver. 1.14.5
(4) Scipy ver. 1.1.0
(5) Pandas ver. 0.23.4
(6) Matplotlib ver.3.0.0
(7) Tensorflow-gpu ver. 1.11.0
(8) CUDA ver. 10.0
(9) NVIDIA Driver ver. 417.22

Our detailed hardware environments are as follows:

(1) Three machines with i9 CPU, each of which is equipped with 2-3 GPUs (GTX 1080 Ti).

To train the three market share prediction models, Model1/2/3, we use the mini-batch size of 2,048 and a learning rate of 1e-4 which decays with a ratio of 0.96 every 100 epochs. We train 1,000 epochs for each model and use the Xavier initializer for initializing weights and the Adam optimizer for updating weights.

For market influence maximization, We have several hyper-parameters, $\bar{\beta}_0$, γ_0, λ, α and so on . All hyper-parameter configurations are already mentioned in the main paper.