Integrated Pricing and Routing on a Network

Pornpawee Bumpensanti, Martin Savelsbergh, He Wang
School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332
pornpawee@gatech.edu, martin.savelsbergh@isye.gatech.edu, he.wang@isye.gatech.edu

We consider an integrated pricing and routing problem on a network. The problem is motivated by applications in freight transportation such as package delivery and less-than-truckload shipping services. The decision maker sets a price for each origin-destination pair of the network, which determines the demand flow that needs to be served. The flows are then routed through the network given fixed arc capacities and costs. Demand for the same origin-destination pair can be routed along multiple paths in the network if desirable. The objective is to maximize the revenues from serving demand minus the transportation costs incurred given the capacity constraints. We propose two algorithms for the solution of this problem: (1) a Frank-Wolfe type algorithm, which requires the objective function to be smooth, and (2) a primal-dual algorithm using an online learning technique, which allows non-smooth objective functions. We prove that the first algorithm has a convergence rate of $O(1/T)$ and the second algorithm has a convergence rate of $O(\log T/T)$, where T is the number of iterations. Numerical experiments on randomly generated instances show that coordinating pricing and routing decisions can improve profits significantly compared to independent pricing or routing strategies.

Key words: transportation, network, pricing, multi-commodity flow

1. Introduction

Most network flow models assume that demands on nodes are exogenous (Ahuja et al. 1993). In this paper, we consider a generalization of the multicommodity network flow problem where demands are determined endogenously by setting prices on node pairs. This setup leads to an intricate interplay between pricing and network routing decisions: on the one hand, pricing decisions determine demand of flows and therefore affect routing decisions; on the other hand, routing decisions affect the cost of serving demand, which in turn informs how to optimally set the prices. In order to maximize overall system profit, the decision maker needs to consider the integration between pricing and routing decisions and optimize them jointly.

We consider a directed network represented by a set of nodes and arcs. Each arc has a fixed capacity and a linear cost function. For each pair of nodes, the decision maker sets a price, which induces a demand that needs to be served. The network may have multiple paths connecting an origin and a destination, so the decision maker chooses how to distribute a demand flow over the different paths, while taking into account the capacity available on the arcs. The goal is to maximize the overall profit for the system, which is determined by the revenues obtained from serving demand minus the costs incurred by transporting demand through the network.

The problem described above has several applications in logistics and freight transportation. For example, consider a package delivery carrier such as FedEx or UPS that operates an expansive service network. The carrier charges prices for packages based on their origins, destinations, and service classes (e.g., next morning, next day, two-day, etc.). The prices will determine the size of demand to be served, since customers are price-sensitive. Because the carrier operates a dense service network, there are possibly multiple options for sending packages from their origins to their destinations. Customers do not necessarily care about which routes their packages take, as long as they are delivered on time. So the carrier can select routes for packages based on transportation costs and capacities. Pricing and routing decisions should ideally be coordinated and made in an integrated manner, since both decisions affect the carrier's profit.

We propose two algorithms for solving the integrated pricing and routing problem. Both algorithms iteratively solves a sub-problem that can be formulated as a minimum cost multicommodity flow (MCMCF) problem. The first algorithm is based on the Frank-Wolfe algorithm (Frank and Wolfe 1956) (also known as the conditional gradient method (Levitin and Polyak 1966)). We show that the rate of convergence of this algorithm is $O(1/T)$ where T is the number of iterations. To obtain this result, the algorithm requires the revenue function to be smooth and concave. The second algorithm is a primal-dual algorithm that updates pricing and routing decisions iteratively. In each iteration, the algorithm executes two steps: the first step is to set prices to maximize profit that adjusted by dual variables; the second step is to determine how to split such demands among multiple paths to minimize cost, as well as updating the dual variables. When the objective function is strongly concave, we show that this primal-dual algorithm has a rate of convergence of $O(\log T/T)$ where T is the number of iterations. An advantage of the primal-dual algorithm over the Frank-Wolfe algorithm is that it allows a non-smooth revenue function (e.g., a piece-wise linear demand function).

Our numerical experiments, using randomly generated instances, show that joint pricing and routing can improve profit by more than 10% compared to making pricing and routing decisions separately. A more in-depth analysis of the resulting solutions shows that in portions of the service network where several demands compete for transport capacity, prices are adjusted to free up capacity for demand with a high profit margin. That is, the price of demand with a low profit margin is raised to reduce that demand and reduce the capacity required to serve it, and the price of demand with a high profit margin is dropped to increase that demand and use the capacity that has become available to serve it. The intricate interactions between price adjustments and routing choices that are revealed demonstrate that optimization techniques are necessary to identify and exploit these opportunities to the fullest.

The remainder of the paper is organized as follows. In Section 2, we review relevant literature. In Section 3, we introduce the integrated pricing and routing problem and give mathematical formulations. In Section 4, we present two algorithms for the solution of the integrated pricing and routing algorithm. In Section 5, we discuss the result of our computational experiments. In Section 6, we offer some final thoughts.

2. Literature Review

Despite the fact that there are numerous papers on network routing problems in which demand is exogenous (e.g., Kennington 1978), there are only a few papers which consider integrated pricing and routing decisions. Mitra et al. (2001) studied a joint pricing and routing problem for revenue maximization in a multi-service network, assuming the set of possible routes is given. Sharkey (2011) studied an integrated pricing and routing problem for a single product, and showed that the problem can be reformulated as a minimum convex cost network flow problem. Lin et al. (2009) and Lin and Lee (2015) formulate joint pricing and routing problems for less-than-truckload transportation as an integer concave programming problem.

The routing problem, i.e., finding origin-destination paths for commodities in a network, is a critical component of Service Network Design (SND) problems. SND integrates capacity and routing decisions, i.e., decisions regarding where and how much capacity to install on the links in a network and decisions regarding the paths that commodities follow from their origin to their destination in the network. For a review of SND, we refer the reader to Crainic (2000) and Wieberneit (2008).

Our proposed methods use the Frank-Wolfe algorithm (Frank and Wolfe 1956, Levitin and Polyak 1966) and the online gradient descent algorithm (Zinkevich 2003). the Frank-Wolfe algorithm under different assumptions on the objective function and feasible set. Zinkevich (2003) shows that the online gradient descent algorithm achieves $O(\sqrt{T})$ regret for a general convex function when using step size $1/\sqrt{t}$ at iteration t. Hazan (2016) provide a stronger regret bound, $O(\log T)$, when running with step size $1/\mu t$ at iteration t for a μ-strongly convex objective function.

Another core component of our methods is the solution of a minimum (linear) cost multi-commodity flow (MCMCF) problem. The MCMCF problems can be found in a wide range of domains, for example, in transportation and logistics (Krile 2004), in communication networks (Resende and Pardalos 2008, Chapter 10) and in scheduling (Vaidyanathan et al. 2007). Even though the MCMCF problem can be formulated as a linear programming and thus solved by a general linear programming solver, the size of instances makes this often inefficient or impractical. As a result, many custom solution methods have developed for solving the MCMCF problem. Comprehensive surveys by Assad (1978) and Kennington (1978) describe many such solution techniques. In addition, a detailed survey of solution approaches for minimum convex cost multicommodity flow problems can be found in Ouorou et al. (2000).

3. Problem Description

We study an integrated pricing and routing problem on a fixed network represented by a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where the set of nodes, $\mathcal{N}$, represents a set of cities and the set of arcs, $\mathcal{E}$, represents connections between pairs of cities. Each arc $e \in \mathcal{E}$ has a per-unit traversal cost c_e and a capacity k_e.

The set of origin-destination (OD) cities for which delivery service is offered is denoted by $\mathcal{J} \subseteq \mathcal{N} \times \mathcal{N}$. We assume that demand is a function of price for each OD pair $j \in \mathcal{J}$ and that this demand function, $d_j(p)$, is known. We also assume that $d_j(p)$ is strictly decreasing, which implies that there is a one-to-one mapping between demands and prices. Let $p_j(d)$ be the inverse demand function of $d_j(p)$, which allows us to recover the price for a given demand. Let $r_j(d) := dp_j(d)$ denote the revenue function, the revenue collected for OD pair $j \in \mathcal{J}$ for demand d. We assume that each revenue function r_j is concave.

The demand of an OD pair $j \in \mathcal{J}$ must be delivered via one or more paths in the network connecting the origin and destination (respecting the capacities of the arcs in the network).

We let $\mathcal{P}_j$ denote the set of possible (directed) paths connecting the origin and destination of OD pair $j \in \mathcal{J}$.

The decision maker wants to maximize the profit, i.e., the total revenue collected minus the total delivery cost incurred. The decision maker can set the price (or, equivalently, demand) and choose the delivery routes for each OD pair, where the arc flows induced by the delivery routes cannot exceed the capacity of the arcs.

Let z_j be the demand of OD pair j. We use x_p to represent the demand delivered using path p and v_e to represent the demand carried on arc e. We let arc-path indicator $\delta_e(p)$ equal to 1 if arc e is in path p and 0 otherwise. We write v, x and z to denote the vectors of v_e, x_p and z_j, respectively. A path-based formulation for the integrated pricing and routing problem is

$$\begin{aligned}
\max_{v,x,z} \quad & \sum_{j\in\mathcal{J}} r_j(z_j) - \sum_{e\in\mathcal{E}} c_e v_e && && \text{(1a)}\\
\text{s.t.} \quad & z_j = \sum_{p\in\mathcal{P}_j} x_p, && \forall j \in \mathcal{J}, && \text{(1b)}\\
& v_e = \sum_{j\in\mathcal{J}} \sum_{p\in\mathcal{P}_j} \delta_e(p) x_p, && \forall e \in \mathcal{E}, && \text{(1c)}\\
& v_e \le k_e, && \forall e \in \mathcal{E}, && \text{(1d)}\\
& x_p \ge 0, && \forall p \in \mathcal{P}_j, j \in \mathcal{J}. && \text{(1e)}
\end{aligned}$$

Constraints (1b) ensure that the demand for an OD pair is delivered using one or more feasible paths for that demand. Constraints (1c) determine the flow on an arc in the network. Constraints (1d) ensure that the flow on an arc does not exceed the arc's capacity. Constraints (1e) ensure that the demand allocated to a path is non-negative. This path-based formulation has $|\mathcal{J}| + \sum_{j\in\mathcal{J}} |\mathcal{P}_j| + |\mathcal{E}|$ decision variables and $|\mathcal{J}| + 2|\mathcal{E}|$ constraints. Note that the number of paths, $|\mathcal{P}| = \sum_{j\in\mathcal{J}} |\mathcal{P}_j|$, is typically large and grows exponentially in the size of network. As the total number of paths becomes prohibitively large even for moderate-size networks, customized solution methods are needed for its solution.

The problem has an equivalent arc-based formulation, which can be found in Appendix A, but our proposed algorithms use the path-based formulation because it is more flexible and can incorporate additional constraints on paths arising in real-world applications. For example, suppose we want to impose constraints on feasible prices such that the prices in the solution do not deviate too much from the current prices. If the new price of OD pair j is restricted to the interval $[\underline{p}_j, \bar{p}_j]$, then, because of the one-to-one mapping between price and demand, the demand z_j has to lie in the interval $[\underline{d}_j, \bar{d}_j]$, where $\underline{d}_j$ and $\bar{d}_j$ are the value of the inverse demand function at $\bar{p}_j$ and $\underline{p}_j$, respectively. More importantly, the path-based formulation can be solved efficiently by the column generation technique (Ford and Fulkerson 1958).

4. Algorithms

In this section, we propose two algorithms to solve the pricing and routing problem. Before we describe these algorithms, we review some definitions for convex functions. We say $\nabla f(x)$ is a subgradient of f at x if for any x_1, it holds that

$$f(x_1) \geq f(x) + \nabla f(x)^\top (x_1 - x).$$

The set of subgradients of f at the point x is called the subdifferential of f at x, and is denoted $\partial f(x)$. If a function f is convex and differentiable, then a subgradient is unique at any x and equal to its gradient at x. A subgradient can exist even when f is not differentiable at x. Note that a subgradient of convex function f always exists. We write $\|x\|$ for the euclidean norm of x, i.e., $\|x\| = \sqrt{x^\top x}$. We say a function f is μ-strongly convex if, for any x_1 and x_2, it holds that

$$f(x_2) \geq f(x_1) + \nabla f(x_1)^\top (x_2 - x_1) + \frac{\mu}{2}\|x_2 - x_1\|^2.$$

Recall that $\nabla f(x)$ a subgradient of f at x and $\partial f(x)$ the subdifferential (i.e., the set of subgradients) of f at x. We say a function g is μ-strongly concave if $-g$ is μ-strongly convex. We say a continuous differentiable function f is β-smooth if

$$f(x_2) \leq f(x_1) + \nabla f(x_1)^\top (x_2 - x_1) + \frac{\beta}{2}\|x_2 - x_1\|^2,$$

which is equivalent to say that a continuous differentiable function f has a β-Lipschitz continuous gradient, i.e.,

$$\|\nabla f(x_1) - \nabla f(x_2)\| \leq \beta \|x_1 - x_2\|.$$

4.1. Frank-Wolfe Algorithm with Column Generation (FW-CG)

The Frank-Wolfe algorithm (Frank and Wolfe 1956), also known as the conditional gradient method (Levitin and Polyak 1966), is a popular first-order method for smooth constrained convex optimization problem of the form $\max\{f(x) \mid x \in X\}$, where f is a concave and smooth function and X is a polyhedron.

At each iteration t, the Frank-Wolfe algorithm evaluates the gradient of function f at current solution x_t, i.e., $\nabla f(x_t)$. Then, it solves a linear programming subproblem given by $w_t = \arg\max_{w\in X} \nabla f(x_t)^\top w$ before updating the solution suing the solution obtained from the linear programming subproblem as $x_{t+1} = x_t + \gamma_t(w_t - x_t)$ where γ_t is step size. The formal description can be found in Algorithm 1. Notice that the Frank-Wolfe algo-

Algorithm 1 The Frank-Wolfe (FW) Algorithm.

```
Input: $x_0 \in X, \gamma_t = 2/(t+2)$ and $T$
for $t = 0, 1, 2, \ldots, T$ do
    Set $w_t \leftarrow \arg\max_{w\in X} \nabla f(x_t)^\top w$
    Set $x_{t+1} \leftarrow x_t + \gamma_t(w_t - x_t)$
Output: $x_T$
```

rithm does not require a projection onto the feasible set but only depends on solving a linear optimization problem over the constrained set. That is, the Frank-Wolfe algorithm performs well if it is inexpensive to solve the linear programming subproblem. The rate of convergence of the Frank-Wolfe algorithm described in Algorithm 1 in the $\beta-$smooth concave objective function is $O(1/T)$ (Lemma 1 in Appendix C, Jaggi 2013, Theorem 1).

When we apply the Frank-Wolfe algorithm to the integrated pricing and routing problem, we solve a convex optimization problem by solving a sequence of linear optimization problems. It is easy to verify that the linear programming subproblem becomes minimum-cost multicommodity flow problem (MCMCF). One way to efficiently solve this linear programming problem is to apply column generation method to path-flow formulation of MCMCF (Ahuja et al. 1993).

Column generation can be used to solve LP with a large number of decision variables. Ford and Fulkerson (1958) outline the idea of using column generation form a maximal multicommodity flow problem, where MCMCF is one of its variation, as follows. The problem is solved by considering two problems: the master problem which includes only a subset of decision variables and the pricing problem which determines if a new variables should be included in the master problem to improve objective function.

Recall that the integrated pricing and routing problem in path-based formulation can be written as

$$\begin{aligned} \max_{x} \quad & \sum_{j\in\mathcal{J}} r_j(\sum_{p\in\mathcal{P}_j} x_p) - \sum_{e\in\mathcal{E}} c_e \sum_{j\in J}\sum_{p\in\mathcal{P}_j} \delta_e(p)x_p \\ \text{s.t.} \quad & \sum_{j\in J}\sum_{p\in\mathcal{P}_j} \delta_e(p)x_p \le k_e, \quad \forall e\in\mathcal{E}, \\ & x_p \ge 0, \quad \forall p\in\mathcal{P}_j, j\in\mathcal{J}\,. \end{aligned} \tag{2}$$

The objective function of (2) is defined as $f(x)$, and a feasible set of (2) is defined as X To apply the Frank-Wolfe algorithm (see Algorithm 1), we first need to compute the gradient of $f(x)$. Each element of the gradient $\nabla f(x)$ is given by, for path $p\in\mathcal{P}_j$,

$$\frac{\partial}{\partial x_p} f(x) = r'_j(\sum_{p\in\mathcal{P}_j} x_p) - \sum_{e\in\mathcal{E}}\sum_{j\in J}\sum_{p\in\mathcal{P}_j} \delta_p(e)c_e.$$

Each iteration, the Frank-Wolfe algorithm solves the following linear programming subproblem.

$$\begin{aligned} \max_{w} \quad & \sum_{j\in\mathcal{J}}\sum_{p\in\mathcal{P}_j} \left[r'_j(\sum_{p\in\mathcal{P}_j} x_{p,t}) - \sum_{e\in\mathcal{E}}\sum_{j\in J}\sum_{p\in\mathcal{P}_j} \delta_p(e)c_e \right] w_p \\ \text{s.t.} \quad & \sum_{j\in J}\sum_{p\in\mathcal{P}_j} \delta_p(e)w_p \le k_e, \quad \forall e\in\mathcal{E}, \\ & w_p \ge 0, \quad \forall p\in\mathcal{P}_j, j\in\mathcal{J}, \end{aligned} \tag{3}$$

where $x_{p,t}$ is the demand delivered on path p obtained in the current iteration t from the Frank-Wolfe algorithm. This linear optimization subproblem can be viewed as the minimum-cost multicommodity flow problem (MCMCF) in path-flow formulation with per-unit cost

$$c_{p,t} = \sum_{e\in\mathcal{E}}\sum_{j\in J}\sum_{p\in\mathcal{P}_j} \delta_e(p)c_e - r'_j(\sum_{p\in\mathcal{P}_j} x_{p,t})$$

on path $p\in\mathcal{P}_j$. It can be observed that the number of decision variables which equals to the number of paths is exponentially large in the network size. To solve this linear programming subproblem, we apply column generation method. The master problem is defined similar to (3) but contains only a subset of paths (decision variables). A new variable is added to the master problem if it can improve the objective function. For maximization problem, the objective function might increase when a new variable with positive reduced cost is included. Therefore, a new variable is included to the master problem when reduced cost is positive. Let $y_{e,t}$ be a dual variable associated with a capacity constraint on arc e at iteration t. Reduced cost of path $p \in \mathcal{P}_j$ can be computed by

$$r_j'(\sum_{p\in\mathcal{P}_j} x_{p,t}) - \sum_{e\in\mathcal{E}} \delta_e(p)(c_e + y_{e,t}).$$

That is, the pricing problem is to check if there exists a path which has positive reduced cost, i.e.,

$$r_j'(\sum_{p\in\mathcal{P}_j} x_{p,t}) - \sum_{e\in\mathcal{E}} \delta_e(p)(c_e + y_{e,t}) > 0 \iff \sum_{e\in\mathcal{E}} \delta_e(p)(c_e + y_{e,t}) < r_j'(\sum_{p\in\mathcal{P}_j} x_{p,t}). \tag{4}$$

If there exists a path which satisfies the condition in (4), we include it to the master problem. Otherwise, the solution is optimal. Such path is easy to find by running shortest path algorithm, e.g. Dijkstra's algorithm (Dijkstra 1959), with modified cost $c_e + y_{e,t}$ on each arc e for each OD pair $j \in \mathcal{J}$ at iteration t. Notice that by apply column generation method all possible paths need not be generated beforehand; instead, path will be generated only as required. The complete definition of the Frank-Wolfe with column generation algorithm (FW-CG) for solving the integrated pricing and routing problem can be found in Algorithm 2.

Before we formally state a theorem, let us recall the definition of big O notation. For two functions $f(T)$ and $g(T) > 0$, we write $f(T) = O(g(T))$ if there exists a constant M_1 and a constant T_1 such that $f(T) \le M_1 g(T)$ for all $T \ge T_1$. The convergence rate of Algorithm 2 follows immediately from Lemma 1 in Appendix C and Theorem 1 in Jaggi (2013) and can be stated as follows.

Theorem 1. *When revenue function is smooth and concave, under Algorithm 2, we have*

$$f(x^*) - f(x_T) \le O\left(\frac{1}{T}\right).$$

4.2. Primal-Dual Algorithm

Next, we propose a Primal-Dual (PD) algorithm. This algorithm uses an iterative method based on applying online learning technique on the Lagrangian relaxation of the original integrated pricing and routing problem. The capacity constraints, which are the only bundle

Algorithm 2 The Frank-Wolfe with Column Generation (FW-CG) Algorithm.

```
Input: $x_0 = 0, y_0 = 0, \gamma_t = 2/(t+2)$ and $T$
Initialize: For each OD pair $j \in \mathcal{J}$, consider a path with minimum cost, i.e., $\mathcal{L}_j = \arg\min_{p\in\mathcal{P}_j} \sum_{e\in\mathcal{E}} \delta_e(p)c_e$.
for $t = 1, 2, \ldots, T$ do
    do
        Set $i \leftarrow 0$
        $(w_t, y_t)$ solves
            $$\begin{aligned} \max_{w} \quad & \sum_{j\in\mathcal{J}}\sum_{p\in\mathcal{L}_j}\left[r'_j(\sum_{p\in\mathcal{L}_j} x_{p,t}) - \sum_{e\in\mathcal{E}}\sum_{j\in J}\sum_{p\in\mathcal{L}_j}\delta_e(p)c_e\right]w_p \\ \text{s.t.} \quad & \sum_{j\in J}\sum_{p\in\mathcal{L}_j}\delta_e(p)w_p \le k_e, \quad \forall e\in\mathcal{E} \quad (y_{e,t}), \\ & w_p \ge 0, \quad \forall p\in\mathcal{L}_j, j\in\mathcal{J}, \end{aligned}$$
        for OD pairs $j \in \mathcal{J}$ do
            if $\min_{p\in\mathcal{P}_j}\sum_{e\in\mathcal{E}}\delta_e^p(c_e + y_{e.t}) < r'_j(\sum_{p\in\mathcal{L}_j} x_t^p)$ then
                Set $\mathcal{L}_j \leftarrow \mathcal{L}_j \cup \arg\min_{p\in\mathcal{P}_j}\sum_{e\in\mathcal{E}}\delta_e^p(c_e + y_{e.t})$
                Set $i \leftarrow i + 1$
    while $i > 0$
    Set $x_{t+1} \leftarrow x_t + \gamma_t(w_t - x_t)$
Output: $x_T$
```

constraint in (2), are removed from the constrained set and placed into the objective function. In each iteration, the algorithm has two steps which can easily be solved. In the first step, it determines total demand, or equivalently price, for each OD pair (pricing decision). Then, it decides how to deliver such demand through one or more paths (routing decision).

In online convex optimization framework, a decision maker chooses x_t at iteration t before seeing a convex cost function ℓ_t. The cost of $\ell_t(x_t)$ is then realized to the decision maker. Many algorithms is proposed with the aim to minimize the regret, where the regret after T iterations is defined as

$$\sum_{t=1}^{T} \ell_t(x_t) - \min_{x \in X} \sum_{t=1}^{T} \ell_t(x). \tag{5}$$

Online Subgradient Descent (Zinkevich 2003) which will be used in the first step of the proposed algorithm is a well-known and simple online learning algorithm. The formal description of the algorithm can be found in Algorithm 3, where $\Pi_X(u)$ is a projection of u onto a convex set X. For a general convex loss function, Zinkevich (2003) shows that

Algorithm 3 The Online Subgradient Descent Algorithm.

```
Input: $x_0 \in X, \eta_t$ and $T$
for $t = 0, 1, 2, \ldots, T$ do
    Set $x_t \leftarrow \Pi_X[x_{t-1} - \eta_t \nabla \ell_t(x_t)]$
```

Algorithm 3 with an appropriate step size has $O(\sqrt{T})$ regret. Hazan (2016) shows that $O(\log T)$ regret is attainable for a strongly convex loss function. This result is formally stated in Lemma 2. Note that the result still holds for a non-smooth convex loss function.

To obtain the Lagrangian dual problem of integrated pricing and routing problem (2), we relax the capacity constraints. Let $y_e \geq 0$ be Lagrange multipliers associated with the capacity constraint of arc e. We let y denote the vector of y_e. The Lagrangian function can be written as

$$L(x,y) = \sum_{j \in \mathcal{J}} r_j(\sum_{p \in \mathcal{P}_j} x_p) - \sum_{e \in \mathcal{E}} c_e \sum_{j \in J} \sum_{p \in \mathcal{P}_j} \delta_e(p) x_p + \sum_{e \in \mathcal{E}} y_e(k_e - \sum_{j \in J} \sum_{p \in \mathcal{P}_j} \delta_e(p) x_p). \tag{6}$$

The Lagrangian dual problem is given by

$$\min_{y \geq 0} \max_{x \geq 0} L(x,y) = L(x^*, y^*) = \max_{x \geq 0} \min_{y \geq 0} L(x,y), \tag{7}$$

where x^* and y^* be the optimal solutions of (7). Because a revenue function is assumed to be concave in a demand for each OD pair $j \in \mathcal{J}$, it follows that the Lagrangian function (6) is concave in total demand z_j of each OD pair $j \in \mathcal{J}$ as well as concave in flow on path x_p for each $p \in \mathcal{P}_j$. Moreover, we can observe that the Lagrangian function (6) is convex (linear) in dual variable y_e for all $e \in \mathcal{E}$.

In the first step of the algorithm, we want to determine the total demand of each OD pair so that the Lagrangian function (6) is maximized. Recall that $z_j = \sum_{p\in\mathcal{P}_j} x_p$. Let z be the vector of z_j for all $j \in \mathcal{J}$. Suppose y is constant. We want to find

$$\begin{aligned} z &= \underset{\substack{z\geq 0 \\ z_j=\sum_{p\in\mathcal{P}_j} x_p}}{\arg\max} \; L(x,y) \\ &= \underset{\substack{z\geq 0 \\ z_j=\sum_{p\in\mathcal{P}_j} x_p}}{\arg\max} \sum_{j\in\mathcal{J}} r_j(z_j) - \sum_{e\in\mathcal{E}} (c_e + y_e) \sum_{j\in\mathcal{J}} \sum_{p\in\mathcal{P}_j} \delta_e(p) x_p. \end{aligned}$$

It is obvious that the problem is separable for each OD pair j, i.e., for all $j \in \mathcal{J}$, we have

$$z_j = \underset{\substack{z_j\geq 0 \\ z_j=\sum_{p\in\mathcal{P}_j} x_p}}{\arg\max} \; r_j(z_j) - \sum_{e\in\mathcal{E}} (c_e + y_e) \sum_{p\in\mathcal{P}_j} \delta_e(p) x_p. \tag{8}$$

That is, z_j is the solution of the profit maximization for each OD pair $j \in \mathcal{J}$. We observe that to obtain z_j the second term in (8) must be minimized. Since the term is basically the modified cost of routing demand of j through paths, all demand must be delivered using only the smallest adjusted cost path. Such path is easily identified by running shortest path algorithm, e.g. Dijkstra's algorithm (Dijkstra 1959) with modified cost $c_e + y_e$ on each arc. Let p_j^* be the shortest cost path of OD pair j with modified cost $c_e + y_e$, i.e., $p_j^* = \arg\min_{p\in\mathcal{P}_j} \sum_{e\in\mathcal{E}} (c_e + y_e)\delta_e(p)$. Therefore, we can write

$$z_j = \arg\max_{z_j \geq 0} r_j(z_j) - c_{p_j^*} z_j, \tag{9}$$

where $c_{p_j^*}$ is the cost of the shortest cost path with modified cost $c_e + y_e$ of OD pair j, i.e., $c_{p_j^*} = \sum_{e\in\mathcal{E}} \delta_e(p_j^*) c_e$. At each iteration the first step applies Online Supergradient Ascent to the maximization problem in (9). That is, for each $j \in \mathcal{J}$, z_j moves in the direction of a supergradient of objective function found in (9).

There are three main underlying reasons why we choose to operate on the total demand on each OD pair instead of the flow on each possible path. Firstly, we do not require to enumerate all possible paths. Secondly, the number of OD pairs is much smaller than the number of all possible paths, which makes the algorithm computationally efficient. Lastly, the algorithm provides a stronger regret guarantee when r_j is assumed to be strongly concave in z_j for all $j \in \mathcal{J}$. This result follows from the result of Online Supergradient Ascent. However, when r_j is assumed to be strongly concave in z_j, it does not imply that r_j is strongly concave in x_p for all $p \in \mathcal{P}_j$ (see more discussion in Section 5.1).

As we mention earlier, the convergence guarantee of Online Supergradient Ascent does not require the Lagrangian function to be smooth in total demand z_j for all $j \in \mathcal{J}$. That is, the result still follows when, for example, price function p_j is piece-wise linear in z_j. This relationship is often observed when there is a price competition.

In the second step of the algorithm, suppose the total demand of each OD pair is fixed from the first step, we want to determine the amount of demand to be delivered on different paths. Based on the Lagrangian dual problem in (7), because z is fixed from the first stage we want to solve

$$\min_{y \geq 0} \max_{\substack{x \geq 0 \\ z_j = \sum_{p \in \mathcal{P}_j} x_j}} L(x, y). \tag{10}$$

We know that

$$\begin{aligned}
\max_{x: \sum_{p \in \mathcal{P}_j} x_p = z_j} L(x,y) = \max_{x} \quad & \sum_{j \in \mathcal{J}} r_j \Big(\sum_{p \in \mathcal{P}_j} x_p\Big) - \sum_{e \in \mathcal{E}} c_e \sum_{j \in J} \sum_{p \in \mathcal{P}_j} \delta_e(p) x_p \\
+ \sum_{e \in \mathcal{E}} y_e \Big(k_e - & \sum_{j \in J} \sum_{p \in \mathcal{P}_j} \delta_e(p) x_p\Big) \\
\text{s.t.} \quad & \sum_{p \in \mathcal{P}_j} x_p = z_j, \quad \forall j \in \mathcal{J}, \\
& x_p \geq 0, \quad \forall p \in \mathcal{P}_j, j \in \mathcal{J},
\end{aligned}$$

and it can be observed that $y_e \geq 0$ for all $e \in \mathcal{E}$ can be viewed as Lagrange multipliers associated with the capacity constraints in the problem in (10). Therefore, x and y can be obtained by solving

$$\begin{aligned}
\max_{x} \quad & \sum_{j \in \mathcal{J}} r_j \Big(\sum_{p \in \mathcal{P}_j} x_p\Big) - \sum_{e \in \mathcal{E}} c_e \sum_{j \in J} \sum_{p \in \mathcal{P}_j} \delta_e(p) x_p \\
\text{s.t.} \quad & \sum_{p \in \mathcal{P}_j} x_p = z_j, \quad \forall j \in \mathcal{J}, \\
& \sum_{j \in J} \sum_{p \in \mathcal{P}_j} \delta_e(p) x_p \leq k_e, \quad \forall e \in \mathcal{E} \quad (y_e \geq 0), \\
& x_p \geq 0, \quad \forall p \in \mathcal{P}_j, j \in \mathcal{J}.
\end{aligned}$$

The first term in objective function, $\sum_{j\in\mathcal{J}} r_j(\sum_{p\in\mathcal{P}_j} x_p) = \sum_{j\in J} r_j(z_j)$, is independent of x, so we can ignore this term. Thus, x and y can be obtained from

$$\begin{aligned} \min_{x} \quad & \sum_{e\in\mathcal{E}} c_e \sum_{j\in J}\sum_{p\in\mathcal{P}_j} \delta_e(p) x_p \\ \text{s.t.} \quad & \sum_{p\in\mathcal{P}_j} x_p = z_j, \qquad \forall j \in \mathcal{J}, \\ & -\sum_{j\in J}\sum_{p\in\mathcal{P}_j} \delta_e(p) x_p \geq -k_e, \quad \forall e\in\mathcal{E} \quad (y_e \geq 0), \\ & x_p \geq 0, \qquad \forall p \in \mathcal{P}_j, j\in\mathcal{J}, \end{aligned} \tag{11}$$

which is the minimum-cost multicommodity flow problem (MCMCF) in path-flow formulation. That is, in the second stage the total demand must be spit in the way that minimize the total cost while satisfying the capacity constraints. Observe that the problem (11) might be infeasible because the demand constraints which force the total demand of each OD pair to be predetermined z_j might cause the violation in capacity constraints. Therefore, we introduce a dummy arc which is uncapacitated for each OD pair to take care residual demand which cannot satisfy the capacity constraints. A large enough per-unit cost of such arc must be set so that the arc will not be used if not necessary.

We can apply the same technique, column generation method, as described in Section 4.1 to solve MCMCF. Specifically, a new variable (path) is added to the master problem when the reduced cost is negative, that is, the pricing problem is to check if for OD pair j there exists a path which satisfies

$$\sum_{e\in\mathcal{E}} \delta_e(p)(c_e + y_e) - \sigma_j < 0 \iff \sum_{e\in\mathcal{E}} \delta_e(p)(c_e + y_e) < \sigma_j, \tag{12}$$

where σ_j is a dual variable associated with the demand constraint of OD pair j. Such path can be found easily by running shortest path algorithm such as Dijkstra's algorithm (Dijkstra 1959) with modified cost $c_e + y_e$ on arc e. Therefore, the MCMCF problem in (11) can be solved efficiently by applying column generation method without having to enumerate all possible path. We state the formal description of PD when objective function is strongly concave in Algorithm 4. Note that we write $\max(x, 0)$ as $[x]^+$.

THEOREM 2. *When revenue function is strongly concave in total demand, under Algorithm 4, we have*

$$\left| \frac{1}{T}\sum_{t=1}^{T} L(x_t, y_t) - L(x^*, y^*) \right| \leq O\left(\frac{\log T}{T}\right).$$

Algorithm 4 The Primal-Dual (PD) Algorithm for Integrated Pricing and Routing.

```
**Input**: $x_0 = 0, y_0 = 0, z_0 = 0, \mu$ and $T$
**Initialize**: For each OD pair $j \in \mathcal{J}$, add an uncapacitated dummy arc with cost $r_j'(0)$ and consider a path with minimum cost, i.e., $\mathcal{L}_j = \arg\min_{p\in\mathcal{P}_j} \sum_{e\in\mathcal{E}} \delta_e(p) c_e$.
**for** $t = 1, 2, \ldots, T$ **do**
    **for** OD pairs $j \in \mathcal{J}$ **do**
        Set $z_{j,t} \leftarrow \left[z_{j,t-1} - \frac{1}{\mu_j t}(\partial - r_j(z_{j,t-1}) + \min_{p\in\mathcal{P}_j} \sum_{e\in\mathcal{E}} (c_e + y_{e,t-1})\delta_e(p))\right]^+.    (13)
    **do**
        Set $i \leftarrow 0$
        $(x_t, y_t)$ solves
            $$\begin{aligned} \min_x \quad & \sum_{e\in\mathcal{E}} c_e \sum_{j\in J} \sum_{p\in\mathcal{P}_j} \delta_e(p) x_p \\ \text{s.t.} \quad & \sum_{p\in\mathcal{P}_j} x_p = z_{j,t}, \quad \forall j \in \mathcal{J}, \\ & -\sum_{j\in J} \sum_{p\in\mathcal{P}_j} \delta_e(p) x_p \geq -k_e, \quad \forall e \in \mathcal{E} \quad (y_e \geq 0), \\ & x_p \geq 0, \quad \forall p \in \mathcal{P}_j, j \in \mathcal{J}, \end{aligned}$$
        **for** OD pairs $j \in \mathcal{J}$ **do**
            **if** $\min_{p\in\mathcal{P}_j} \sum_{e\in\mathcal{E}} \delta_e(p)(c_e + y_{e,t}) < \sigma_j$ **then**
                Set $\mathcal{L}_j \leftarrow \mathcal{L}_j \cup \arg\min_{p\in\mathcal{P}_j} \sum_{e\in\mathcal{E}} \delta_e(p)(c_e + y_{e,t})$
                Set $i \leftarrow i + 1$
    **while** $i > 0$
**Output**: $\bar{x} = \sum_{t=1}^T x_t / T$ and $\bar{y} = \sum_{t=1}^T y_t / T$
```

Appendix B.1 provides the detailed proof.

THEOREM 3. *When revenue function is strongly concave in total demand, under Algorithm 4, we have*

$$\|\bar{z} - z^*\|^2 \leq O\left(\frac{\log T}{T}\right).$$

The proof can be found in Appendix B.2.

We remark that unlike the Frank-Wolfe algorithm, the PD algorithm (Algorithm 4) does not require the objective function to be smooth. We obtain the same rates of convergence in Theorem 2 and Theorem 3 for strongly concave and both smooth and non-smooth objective function. Moreover, the PD algorithm can be applied even if the objective function is not strongly concave. In that case, Equation (13) in Algorithm 4 needs to be modified. Specifically, for general concave (not necessarily strongly concave) objective function, Equation (13) can use a step size of $1/\sqrt{t}$ in place of the step size $1/\mu_j t$. However, for general concave objective, we can only conclude that rate of convergence in term of average regret of this algorithm is $O(1/\sqrt{T})$.

To summarize this section, we proposed two algorithms, Frank-Wolfe with Column Generation (FW-CG) and primal-dual algorithm (PD), for solving integrated pricing and routing problem. When revenue function is smooth and concave, FW-CG achieves the rate of convergence in term of objective function of $O(1/T)$. However, PD still works without a smoothness assumption. We show that for strongly concave revenue function the rates of convergence of PD in term of average regret and total demand are both $O(\log T/T)$. We note that we can alter PD so that the algorithm allows general concave revenue function. Specifically, step size in the first step when updating total demand of each OD pair needs to be modified. At iteration t, modified PD uses step size $1/\sqrt{t}$ instead of $1/\mu_j t$. This modified algorithm with new step size has the rate of convergence in term of average regret of $O(1/\sqrt{T})$ for general concave revenue function. The proof is similar to The proof of Theorem 2, but we use $O(\sqrt{T})$ bound (Zinkevich 2003) when we apply Online Supergradient Ascent. Note that when revenue function is not strongly concave, we are not able to conclude the rate of convergence in term of total demand.

5. Numerical Experiments

In the experiments below, we consider a service network with 10 cities and connections between all cities, i.e., a complete directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ with $|\mathcal{N}| = 10$ and $|\mathcal{E}| = 90$, resulting in $|\sum_{j\in J} P_j| = 9,864,090$ paths. Let $U\{\underline{u}, \bar{u}\}$ denote the discrete uniform distribution which can have integer values from $\underline{u}$ and $\bar{u}$. The per-unit traversal cost and the capacity are drawn from $U\{1,5\}$ as well for all arcs $e \in \mathcal{E}$, i.e., $c_e \sim U\{1,5\}$ and $k_e \sim U\{1,5\}$.

5.1. Demand Models

We consider two different types of demand models.

5.1.1. Linear Demand Model. We assume linear demand models for all origin-destination (OD) pairs $j \in J$ with different parameter values:

$$d_j(p) = a_j - b_j p,$$

where $a_j > 0$ and $b_j > 0$. Equivalently, we can write the price as a linear function of demand as

$$p_j(d) = \frac{a_j}{b_j} - \frac{1}{b_j} d.$$

Recall that the revenue function $r_j(d) := p_j(d)d$ requires to be concave. For linear demand model, we can then write

$$r_j(d) = \frac{a_j}{b_j} d - \frac{1}{b_j} d^2,$$

which is strongly concave in d, because $r_j''(d) = -2/b_j < 0$. Note that although revenue function of linear demand model is strongly concave in total demand, it is concave but not strongly concave in the flow on each path. To see this, we write total demand as a summation of flows on each possible path. We have $\nabla^2 r_j(d) = -2J_{|\mathcal{P}_j|}$, where J_n is an $n \times n$ matrix of ones. Its eigenvalues are $-2|\mathcal{P}_j|$ and 0, which are less than or equal to zero. Therefore, revenue function of linear demand model is concave but not strongly concave in flow on each path. We assume that the slope parameter of the price function is drawn from $U\{1,5\}$ for all OD pairs $j \in \mathcal{J}$, i.e., $1/b_j \sim U\{1,5\}$.

5.1.2. Piece-wise Linear Demand Model. We assume that, for each OD pair $j \in J$, the demand function can be written as

$$d_j(p) = \begin{cases} a_{j,1} - b_{j,1}p & p \le p_0 \\ a_{j,2} - b_{j,2}p & p > p_0, \end{cases}$$

where $a_{j,1} > 0, a_{j,2} > 0, b_{j,1} > 0$ and $b_{j,2} > 0$. Equivalently, we can write the price as a piece-wise linear function of demand as

$$p_j(d) = \begin{cases} \frac{a_{j,1}}{b_{j,1}} - \frac{1}{b_{j,1}}d & d \le d_0 \\ \frac{a_{j,2}}{b_{j,2}} - \frac{1}{b_{j,2}}d & d > d_0. \end{cases}$$

Specifically, we are interested in the demand function of the following form.

$$p_j(d) = \begin{cases} \frac{a_j}{b_j} - \frac{1}{b_j} d & d \le \frac{a_j}{4} \\ \frac{5a_j}{4b_j} - \frac{2}{b_j} d & d > \frac{a_j}{4}. \end{cases}$$

The revenue function can then be written as

$$r_j(d) = \begin{cases} \frac{a_j}{b_j} d - \frac{1}{b_j} d^2 & d \le \frac{a_j}{4} \\ \frac{5a_j}{4b_j} d - \frac{2}{b_j} d^2 & d > \frac{a_j}{4}. \end{cases}$$

It can be seen that this revenue function is not differentiable at $a_j/4$. The subdifferential of revenue function is

$$\partial r_j(d) = \begin{cases} \frac{a_j}{b_j} - \frac{2}{b_j} d & d < \frac{a_j}{4} \\ \left[\frac{a_j}{2b_j}, \frac{a_j}{4b_j}\right] & d = \frac{a_j}{4} \\ \frac{a_j}{b_j} - \frac{4}{b_j} d & d > \frac{a_j}{4}, \end{cases}$$

which is decreasing function in d. Therefore, the revenue function is concave. However, because the revenue function is non-smooth, only PD is applicable to this setting. We assume that the slope parameter of the price function is drawn from $U\{1,5\}$ for all OD pairs $j \in \mathcal{J}$, i.e., $1/b_j \sim U\{1,5\}$.

5.2. Experimental Results

We test four different distributions, $U\{7,11\}$, $U\{9,13\}$, $U\{11,15\}$, and $U\{13,17\}$ for the intercept parameter of the price function, a_j/b_j. Consequently, the same demand can be induced by setting a higher price, and higher profit is obtained for the same demand. We generate three instances for each distribution of the intercept parameter of price function while assuming other parameters have the same distributions.

We investigate the benefit of integrating pricing and routing decisions by considering three different settings:

1. Pricing only: The decision maker sets optimal prices for each OD pair given that the demand for the OD pair only uses the path consisting of the direct arc that links the origin to the destination.

2. Pricing → Routing: The decision maker sets optimal prices for each OD pair assuming that the demand for the OD pair only uses the path consisting of the direct arc that links the origin to the destination, but after setting the prices, the decision maker optimally chooses one or more routes for each OD pair to serve the resulting demand.

3. PRICING + ROUTING: The decision maker simultaneously and optimally determines the prices and the routes to serve the resulting demand for each OD pair, i.e., solve the integrated pricing and routing problem.

5.2.1. Linear Demand Model. Figure 1 shows the average profit for the three settings when the intercept term of the price function, a_j/b_j, is drawn from $U\{7,11\}$, $U\{9,13\}$, $U\{11,15\}$, and $U\{13,17\}$. We observe a slight increase in profit, 5.23%, 3.17%, 1.87%, and 1.08%, respectively, for PRICING → ROUTING over PRICING ONLY, and a significant increase in profit, 11.39%, 10.96%, 11.08%, and 11.54%, respectively, for PRICING + ROUTING over PRICING ONLY.

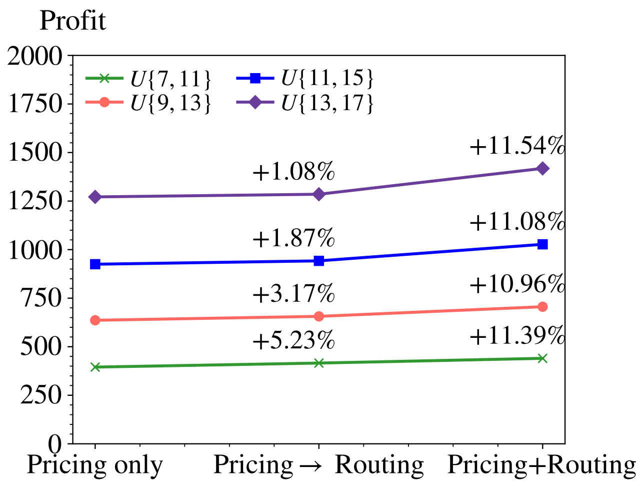

Figure 1 **The average profit of Pricing only, Pricing → Routing and Pricing + Routing when the intercept term of the price function, a_j/b_j, is drawn from $U\{7,11\}$, $U\{9,13\}$, $U\{11,15\}$ and $U\{13,17\}$.**

We also see that the improvement in average profit of PRICING → ROUTING over PRICING ONLY decreases as the intercept term of price function increases. The reason is that the optimal demand for each OD pair never decreases (and likely increases) when the intercept term of price function increases. As a result, the capacity utilization increases which restricts the allocation of demand across the different paths when deciding the routes for the demands.

Figure 2 shows the average arc capacity utilization in the PRICING ONLY, PRICING → ROUTING, and PRICING + ROUTING solutions when the intercept term of the price function, a_j/b_j, is drawn from $U\{7,11\}$, $U\{9,13\}$, $U\{11,15\}$, and $U\{13,17\}$. This graph shows that effectively using the capacity in the service network is critical to achieving high profits. Even though the demand is the same in the Pricing only and Pricing $\rightarrow$ Routing settings, the arc capacity utilization increases because some of the demand is allocated to longer, but cheaper, alternative paths.

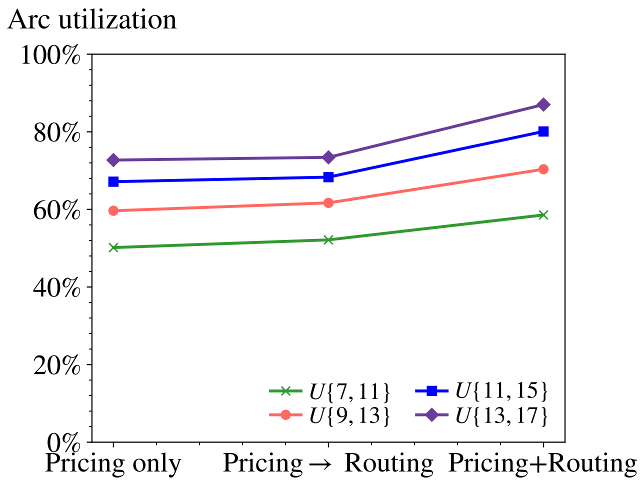

Figure 2 **The average capacity utilization of Pricing only, Pricing $\rightarrow$ Routing and Pricing + Routing when demand function is linear, and the intercept term of the price function, a_j/b_j, is drawn from $U\{7,11\}$, $U\{9,13\}$, $U\{11,15\}$ and $U\{13,17\}$.**

Figure 1 clearly shows that simultaneously optimizing pricing and routing decisions pays off. In the Pricing + Routing solution, the price for an OD pair often differs from the price for the same OD pair in the Pricing only solution to induce more demand for more profitable OD pairs and less demand to less profitable OD pairs. Two factors impact the profit that can be achieved when simultaneously optimizing pricing and routing decisions: (1) the number profitable paths, i.e., paths for which the per-unit revenue is greater than the per-unit cost, and (2) the capacity utilization. As mentioned earlier, an increase in the intercept term of the price function increases the capacity utilization, which, in turn, reduces the routing flexibility. The larger the number of profitable paths for an OD pair, the more options can be exploited by the optimization.

Figure 3 shows the average number of paths for which the per-unit revenue is greater than the per-unit cost when using the prices set in the Pricing only solution, when the intercept term of price function a_j/b_j is drawn from $U\{7,11\}$, $U\{9,13\}$, $U\{11,15\}$, and $U\{13,17\}$. We see that the average number of profitable paths increases exponentially

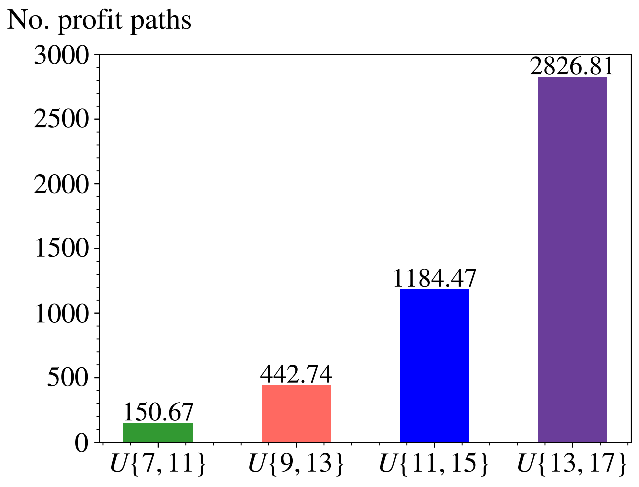

Figure 3 **The average number of profitable paths when using the prices of the Pricing only solution when demand function is linear, and the intercept term of price function a_j/b_j is drawn from $U\{7,11\}$, $U\{9,13\}$, $U\{11,15\}$ and $U\{13,17\}$.**

when the intercept term of the price function increases.

The interaction between these two factors has different effects in different instances, which explains why we do not see a monotonic change in average profit when the intercept term of the price function is increased.

Next, we investigate the solution to a single instance in more depth (the intercept term of price function for this instance is drawn from $U\{11,15\}$). We focus on portion of the service network consisting of four nodes and five arcs; the arcs with their cost and capacity are shown in Figure 4.

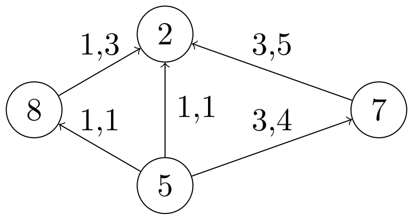

Figure 4 **Cost (first element) and capacity (second element) of a subset of arcs in the network**

In Table 1, we show, for each of the five OD pairs corresponding to the five arcs, the size of the demand, along how many paths the demand is routed, and the profit obtained from serving the demand in the optimal solution for the Pricing only and for the Pricing + Routing setting. Furthermore, in Table 2, we show, for each of the arcs, the total demand on the arc and the arc's capacity utilization in the optimal solution for the Pricing only and for the Pricing + Routing setting as well as the specific demands allocated to the arc and their contribution to the arc's capacity utilization.

Table 1 Demands of a subset of OD pairs in Pricing only and Pricing + Routing

OD	Pricing only			Pricing + Routing				
	price	demand	total profit	price	demand	#routes	avg per-unit profit	total profit
(5,2)	10.00	1.00	9.00	9.22	1.39	3	7.87	10.93
(5,7)	8.00	1.00	5.00	8.00	1.00	1	5.00	5.00
(5,8)	6.00	1.00	5.00	7.82	0.64	1	6.82	4.33
(7,2)	9.00	1.50	9.00	9.22	1.44	1	6.22	8.99
(8,2)	7.00	1.50	9.00	7.40	1.40	1	6.40	8.96

We see in Table 1 that the demand for an OD pair in the solution for the Pricing + Routing setting can go up or down when compared to the demand in the solution for the Pricing only setting. Because in the Pricing + Routing setting, the demand can be delivered using multiple paths, the OD pairs are competing for the capacity on the arcs. Therefore, in an optimal solution, the capacity should be used by the more profitable OD pairs. In fact, for OD pairs with a high per-unit profit, it may be beneficial to induce more demand, by setting a lower price, while for OD pairs with low per-unit profit, it may be beneficial to induce less demand, by setting a higher price, in order to create capacity that can be exploited by more profitable OD pairs. For example, in the solution for the Pricing + Routing setting, the demand for OD pair (5,2), which has a high per-unit profit, is increased and delivered using the paths $5 \to 8 \to 2$ and $5 \to 7 \to 2$. The demands of OD pairs (5,8), (5,7), and (7,2), which have a low per-unit profit, are decreased to free up capacity for the demand of OD pair (5,2).

In the Pricing + Routing setting, where demand can be satisfied using multiple paths, the arc capacity has to be shared by OD pairs. Because different OD pairs have different price functions, it becomes virtually impossible to construct solutions with high profit solutions without the use of optimization. To illustrate, the demand of OD pair (5,2) is 1.0 in the solution to the Pricing only setting, which uses the entire capacity of arc (5,2). If the arc (5,2) would have been uncapacitated, the optimal demand for OD pair (5,2) in

Table 2 Flows and utilization of a subset of arcs in Pricing only and Pricing + Routing

arc	Pricing only			Pricing + Routing		
	OD	flow	utilization	OD	flow	utilization
5→2		1.00	1.00		1.00	1.00
	(5,2)	1.00	1.00	(5,2)	1.00	1.00
5→7		1.00	0.25		1.02	0.26
	(5,7)	1.00	0.25	(5,2)	0.02	0.01
				(5,7)	1.00	0.25
5→8		1.00	1.00		1.00	1.00
	(5,8)	1.00	1.00	(5,2)	0.36	0.36
				(5,8)	0.64	0.64
7→2		1.50	0.30		5.00	1.00
	(7,2)	1.50	0.30	(5,2)	0.02	0.00
				(7,2)	1.44	0.29
				(7,0)	*0.25*	*0.05*
				(0,2)	*0.46*	*0.09*
				(7,8)	*2.82*	*0.56*
8→2		1.50	0.50		3.00	1.00
	(8,2)	1.50	0.50	(5,2)	0.36	0.12
				(8,2)	1.40	0.47
				(0,2)	*0.49*	*0.16*
				(1,2)	*0.52*	*0.17*
				(8,1)	*0.23*	*0.08*

the Pricing only setting would have been 2.75. In the Pricing + Routing setting, the optimal demand for OD pair (5,2) is no longer restricted (only) by the capacity of arc (5,2). Therefore, the price for OD pair (5,2) is reduced from 10.00 to 9.22 to induce a higher demand of 1.39. This is still much less than 2.75, because (1) the costs of the alternative paths, and (2) the competition for capacity on the arcs on the alternative paths. In the solution for the the Pricing + Routing setting, the demand of OD pair (5,2) is split among paths 5→2 (1 unit), 5→8→2 (0.36 units), and 5→7→2 (0.02 units). The per-unit cost of path 5→2 is 1.0, while the per-unit costs of the paths 5→8→2 and 5→7→2 are 2 and 6, respectively; these paths are clearly more expensive. Since 0.36 of the capacity of arc 5→8 is allocated to the demand of OD pair (5,2), the demand of OD pair (5,8), which in the solution for PRICING ONLY setting used the entire capacity of arc 5 → 8, is decreased to 0.64 by increasing its price from 6.00 to 7.82.

5.2.2. Piece-wise Linear Demand Model. Figure 5 shows the average profit for the three settings when the demand function is piece-wise linear and the intercept term of the price function, a_j/b_j, is drawn from $U\{7,11\}$, $U\{9,13\}$, $U\{11,15\}$, and $U\{13,17\}$. Similar to linear demand case, we observe a slight increase in profit, 6.62%, 4.36%, 2.86%, and 2.01%, respectively, for PRICING → ROUTING over PRICING ONLY, and a significant increase in profit, 12.95%, 12.79%, 13.41%, and 14.38%, respectively, for PRICING + ROUTING over PRICING ONLY.

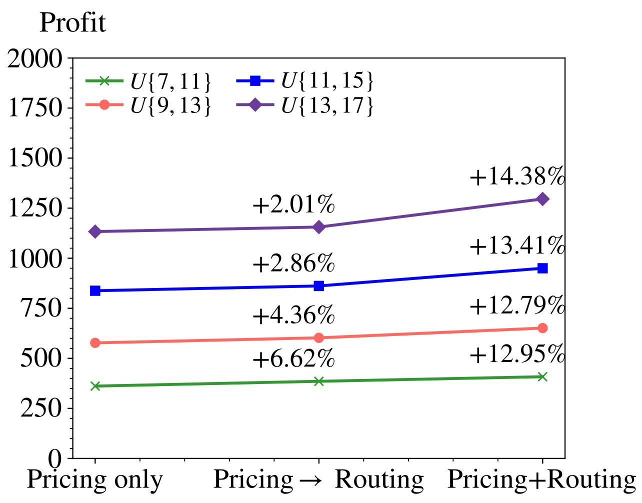

Figure 5 **The average profit of Pricing only, Pricing → Routing and Pricing + Routing when demand function is piece-wise linear and the intercept term of the price function, a_j/b_j, is drawn from $U\{7,11\}$, $U\{9,13\}$, $U\{11,15\}$ and $U\{13,17\}$.**

Figure 6 shows the average arc capacity utilization in the PRICING ONLY, PRICING → ROUTING, and PRICING + ROUTING solutions when the demand function is piece-wise linear and the intercept term of the price function, a_j/b_j, is drawn from $U\{7,11\}$, $U\{9,13\}$, $U\{11,15\}$, and $U\{13,17\}$. Figure 7 shows the average number of paths for which the per-unit revenue is greater than the per-unit cost when using the prices set in the PRICING ONLY solution, when the demand function is piece-wise linear and the intercept term of price function a_j/b_j is drawn from $U\{7,11\}$, $U\{9,13\}$, $U\{11,15\}$, and $U\{13,17\}$.

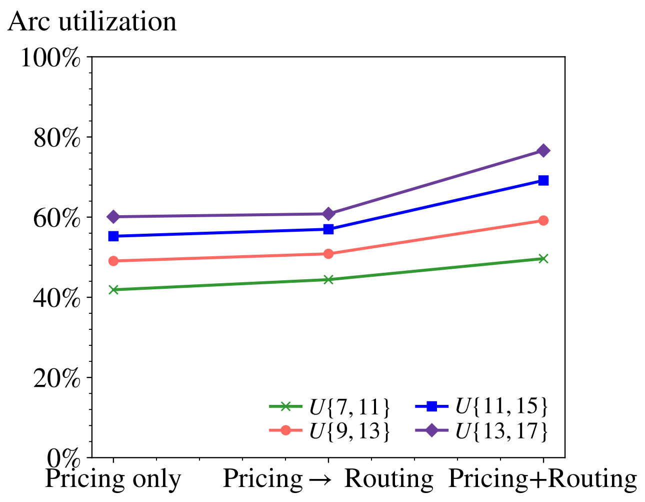

Figure 6 **The average capacity utilization of Pricing only, Pricing $\rightarrow$ Routing and Pricing + Routing when demand function is piece-wise linear and the intercept term of the price function, a_j/b_j, is drawn from $U\{7,11\}$, $U\{9,13\}$, $U\{11,15\}$ and $U\{13,17\}$.**

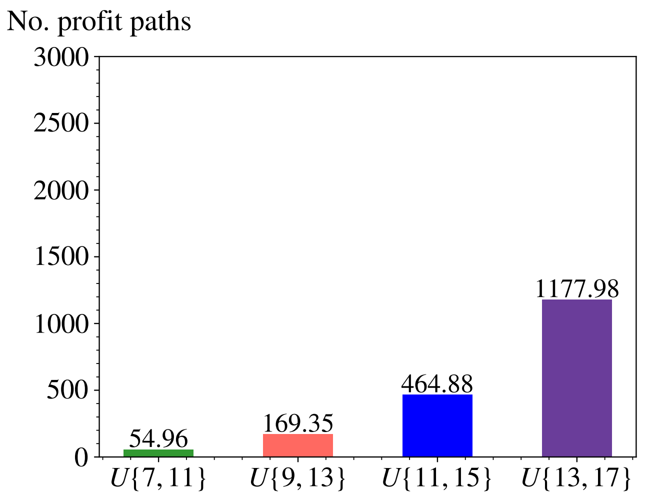

Figure 7 **The average number of profitable paths when using the prices of the Pricing only solution when the intercept term of price function a_j/b_j is drawn from $U\{7,11\}$, $U\{9,13\}$, $U\{11,15\}$ and $U\{13,17\}$.**

As discussed in Section 5.2.1, because of limited arc capacity, we observe a monotonic decrease in average profit improvement of PRICING $\rightarrow$ ROUTING over PRICING ONLY when the intercept term of the price function increases. However, because of the interaction between the capacity utilization and the number of profitable paths, we cannot observe such trend in average profit improvement of PRICING + ROUTING over PRICING ONLY.

5.3. Computation Time and Convergence Rates.

In this section, we will study the convergence rate to the optimal solution of the Frank-Wolfe algorithm with column generation (FW-CG) and the Primal-Dual algorithm (PD) when applicable. We test one problem instance with the intercept term of the price function a_j/b_j drawn from $U\{11, 15\}$.

We apply the proposed algorithms, when applicable, to the simulated network for 10000 iterations. We measure performance of the algorithms using the metrics found in Table 3. The first metric, log loss, measures the gap between objective value obtained from proposed algorithm and optimal objective value. The second metric, log optimal demand gap measures the total difference between demand solutions obtained from proposed algorithm and optimal demands. The third metric, log optimal path flow gap measures the total difference between path flow solutions obtained from proposed algorithm and optimal path flows, and the last metric, log optimal arc flows gap, measures the total difference between arc flow solutions obtained from proposed algorithm and optimal arc flows.

Table 3 The performance metrics.

Metric	FW-CG	PD
log loss	$\log(f(x^*) - f(x_t))$	$\log \lvert \frac{1}{t}\sum_{s=1}^{t} L(x_s, y_s) - L(x^*, y^*) \rvert$
log optimal demand gap	$\log \lVert z_t - z^* \rVert^2$	$\log \lVert \bar{z}_t - z^* \rVert^2$
log optimal path flow gap	$\log \lVert x_t - x^* \rVert^2$	$\log \lVert \bar{x}_t - x^* \rVert^2$
log optimal arc flows gap	$\log \lVert v_t - v^* \rVert^2$	$\log \lVert \bar{v}_t - v^* \rVert^2$

5.3.1. Linear Demand Model.

Recall that we assume that each OD pair has a linear relationship between demand and price. That is, the revenue function is both smooth and strongly concave. Therefore, both the Frank-Wolfe algorithm with column generation (FW-CG) and the Primal-Dual algorithm (PD) can be applied in order to solve the integrated pricing and routing problem.

We compare the convergence results of two algorithms when measured against log iteration across four metrics in Figure 8. Note that we will suppress base 10 of logarithmic function and write log to represent $\log_{10}$ in this section.

Figure 8 shows that log loss and log iteration has linear relationships with negative slope for both algorithms. This numerical result coincides with the theoretical results in Lemma

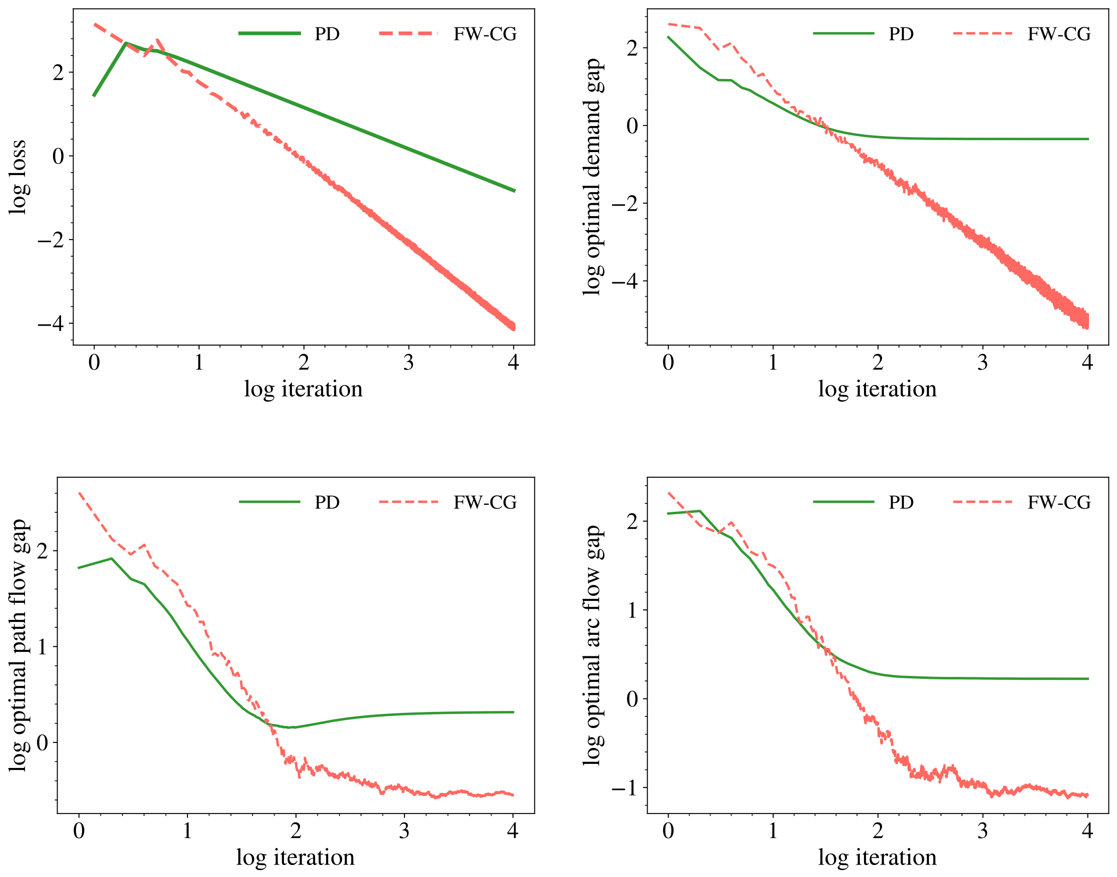

Figure 8 **The plots compare** log **loss (upper left),** log **optimal demand gap (upper right),** log **optimal path flow gap (lower left) and** log **optimal arc flows gap (lower right) of PD and FW-CG against** log **iteration (x-axis) when demand function is linear.**

1 and Theorem 2. Moreover, we can observe that FW-CG has a faster convergence rate in term of objective value than PD. Note that although FW-CG converges faster, it requires an additional condition, smoothness, on revenue function, while PD only needs revenue function to be strongly concave. Even though we only show the convergence of demand when using PD in Theorem 3, Figure 8 show numerically the convergence of demand, path flow and arc flow from both proposed algorithms. In fact, we can observe that they seem to converge in the same order. When we examine the run time, PD and FW-CG take 96.93 seconds and 67.65 seconds per 10000 iterations respectively.

Next, we show how sensitive of running time to an increase in the number of nodes of both algorithms. Figure 9 plots average time per iterations (seconds) used by PD and FW-CG when demand function is linear and the number of nodes is $10, 20, \ldots, 50$. It can be seen that the running time of PD increases exponentially as the number of nodes increases, while

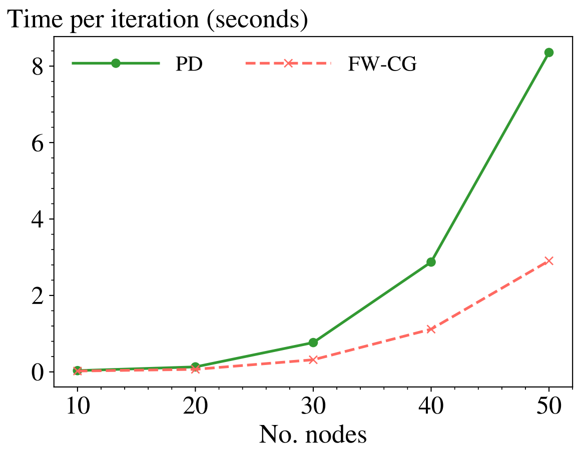

Figure 9 **The plot shows average time per iteration (seconds) used by PD and FW-CG when demand function is linear and the number of nodes is $10, 20, \ldots, 50$.**

the running time of FW-CG seems to increase linearly as the number of nodes increases. One reason why we observe larger running time in PD is because linear programming subproblems of PD require more time to solve than those of FW-CG due to the number of decision variables generated from column generation method (See Figure 10).

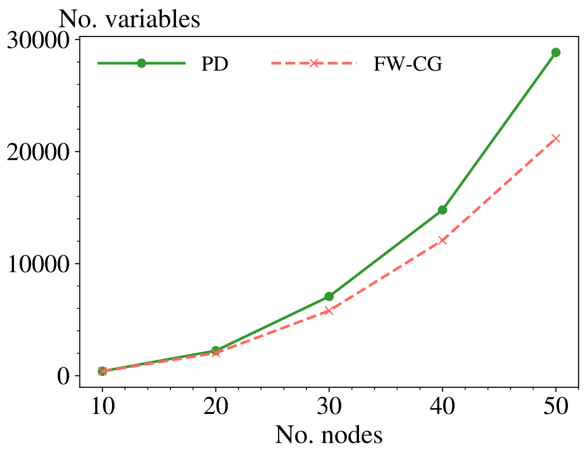

Figure 10 **The plot shows the number of decision variables involved in linear programming subproblem of PD and FW-CG when demand function is linear and the number of nodes is $10, 20, \ldots, 50$.**

5.3.2. Piece-wise Linear Demand Model. Recall that when we assume demand function to be piece-wise linear, we can show that the revenue function is concave but non-smooth. Since the Frank-Wolfe algorithm with column generation (FW-CG) requires smooth objective function, only the Primal-Dual algorithm (PD) can be applied to solve this problem.

We run the Primal-Dual algorithm (PD) on the simulated network for 10000 iterations. The convergence results can be found in Figure 11. It can be seen that the slope of log loss and log optimal demand gap in Figure 11 scale linearly with log(iteration) which are consistent with Theorem 2 and Theorem 3. When we examine the run time, PD takes 92.95 seconds per 10000 iterations.

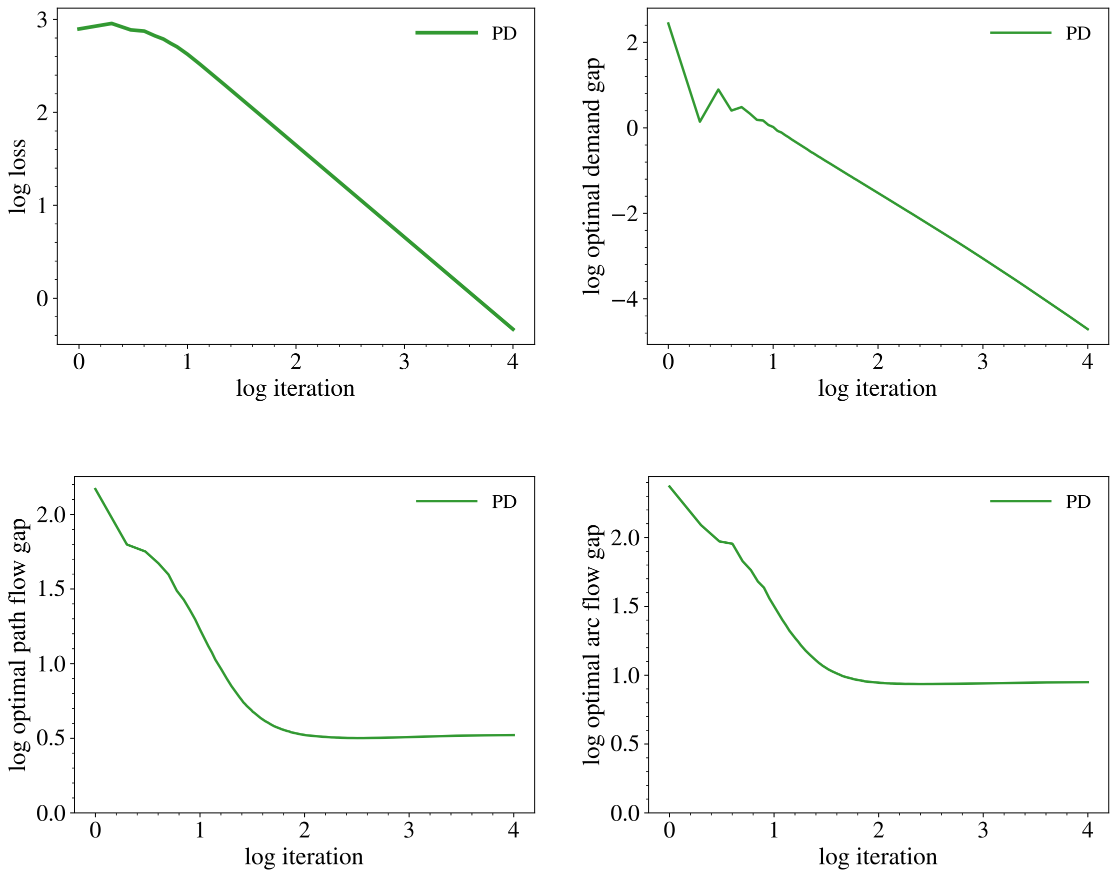

Figure 11 **The plots compare** log **loss (upper left),** log **optimal demand gap (upper right),** log **optimal path flow gap (lower left) and** log **optimal arc flows gap (lower right) of PD against** log **iteration (x-axis) when demand function is piece-wise linear.**

6. Final Remarks

We have studied an integrated pricing and routing problem on a service network. The problem can be formulated as a convex optimization problem, although the size of this optimization problem prohibits us from solving it directly. We have proposed two algorithms to solve the problem. First, we modified the classical Frank-Wolfe algorithm with column generation (FW-CG). When the objective function is smooth, we show that the rate of convergence in terms of the objective function is $O(1/T)$. Second, we propose a primal-dual algorithm (PD), which allows a non-smooth objective function (e.g., a piece-wise linear pricing function). We show that when the objective is strongly concave, the rate of convergence of PD in terms of average regret is $O(\log T/T)$. Numerical experiments demonstrate the benefit of joint pricing and routing; it can increase profit by more than 10%.

There are many possible directions for future research. We mention only two here. We have assumed that the service capacity in the service network is known. A natural extension is to let the optimization decide whether it is beneficial to acquire additional capacity on some of the links in the network. This can be done by allowing demand on an arc to exceed capacity at a cost. Another natural extension is to consider convex arc cost, e.g., per unit cost is an increasing step function as a result of resource scarcity. In this case, we can apply PD to solve the problem; however, the sub-problem becomes a nonlinear MCMCF problem where the objective function is piece-wise linear.

Appendix

A. Arc-based Formulation

Let $v_{j,e}$ be the demand of OD pair j on arc e for all $j \in \mathcal{J}$ and $e \in \mathcal{E}$. Let S_j and D_j denote the origin and destination associated with OD pair j for all $j \in \mathcal{J}$. Furthermore, let $\mathcal{E}^-(i)$ and $\mathcal{E}^+(i)$ denote the set of incoming and outgoing arcs of node i, respectively. The integrated pricing and routing problem can also be formulated as

$$\max_{v,x,z} \quad \sum_{j\in\mathcal{J}} r_j(z_j) - \sum_{e\in\mathcal{E}} c_e v_e \tag{14a}$$

$$\text{s.t.} \quad \sum_{e\in\mathcal{E}^+(i)} v_{j,e} - \sum_{e\in\mathcal{E}^-(i)} v_{j,e} = \begin{cases} z_j & \text{for } i = S_j \\ 0 & \text{for } i \in \mathcal{N}, i \neq S_j, D_j, \forall j \in \mathcal{J}, \\ -z_j & \text{for } i = D_j \end{cases} \tag{14b}$$

$$v_e = \sum_{j\in\mathcal{J}} v_{j,e}, \qquad \forall e \in \mathcal{E}, \tag{14c}$$

$$v_e \le k_e, \qquad \forall e \in \mathcal{E}, \tag{14d}$$

$$v_e \ge 0, \qquad \forall e \in \mathcal{E}. \tag{14e}$$

Constraints (14b) ensure demand is delivered from origin S_j to destination D_j for all OD pairs $j \in \mathcal{J}$. Constraints (14c) determine the total flow on each arc in the network. Constraints (14d) enforce the capacity restrictions for the arcs. Constraints (14e) ensure that the demand allocated to an arc is non-negative. The arc-based formulation has $|\mathcal{J}|+|\mathcal{E}|+|\mathcal{J}||\mathcal{E}|$ decision variables and $|\mathcal{N}||\mathcal{J}|+2|\mathcal{E}|$ constraints.

B. Proofs of Theorems

B.1. Proof of Theorem 2

Proof of Theorem 2 Define $\tilde{L}(z,y)=\max_{\substack{z\geq 0\\ z_j=\sum_{p\in\mathcal{P}_j}x_p}} L(x,y)$. It follows that

$$\begin{aligned}\max_{z\geq 0}\sum_{t=1}^{T}\tilde{L}(z,y_t) &= \max_{z\geq 0}\sum_{t=1}^{T}\max_{\substack{z\geq 0\\ z_j=\sum_{p\in\mathcal{P}_j}x_p}} L(x,y_t)\\ &= \max_{z\geq 0}\max_{\substack{z\geq 0\\ z_j=\sum_{p\in\mathcal{P}_j}x_p}}\sum_{t=1}^{T} L(x,y_t)\\ &= \max_{x\geq 0}\sum_{t=1}^{T} L(x,y_t).\end{aligned}$$

Because we apply Online Supergradient Ascent on a sequence of $\tilde{L}(z,y_t)$ which is strongly concave in z, from Lemma 2, we have

$$\begin{aligned}O(\log T) &\geq \max_{z\geq 0}\sum_{t=1}^{T}\tilde{L}(z,y_t)-\sum_{t=1}^{T}\tilde{L}(z_t,y_t)\\ &= \max_{x\geq 0}\sum_{t=1}^{T}L(x,y_t)-\sum_{t=1}^{T}\tilde{L}(z_t,y_t).\end{aligned} \tag{15}$$

We have shown previously that x and y which solve MCMCF in (11) are identical to x and y which solve $\min_{y\geq 0}\max_{\substack{x\geq 0\\ \sum_{p\in\mathcal{P}_j}x_p=z_j}} L(x,y)$. Therefore, from definition of x_t and y_t in Algorithm 4, we have

$$\begin{aligned}L(x_t,y_t) &= \min_{y\geq 0}\max_{\substack{x\geq 0\\ \sum_{p\in\mathcal{P}_j}x_p=z_{j,t}}} L(x,y) = \max_{\substack{x\geq 0\\ \sum_{p\in\mathcal{P}_j}x_p=z_{j,t}}}\min_{y\geq 0} L(x,y)\\ &= \min_{y\geq 0} L(x_t,y) = \max_{\substack{x\geq 0\\ \sum_{p\in\mathcal{P}_j}x_p=z_{j,t}}} L(x,y_t).\end{aligned} \tag{16}$$

By definition of $\tilde{L}$, we know that

$$\tilde{L}(z_t,y_t) = \max_{\substack{x\geq 0\\ \sum_{p\in\mathcal{P}_j}x_p=z_{j,t}}} L(x,y_t) = L(x_t,y_t). \tag{17}$$

Hence, from (15), it follows that

$$O(\log T) \geq \max_{x\geq 0}\sum_{t=1}^{T}L(x,y_t)-\sum_{t=1}^{T}L(x_t,y_t)$$

$$\geq \sum_{t=1}^{T}L(x^*,y_t)-\sum_{t=1}^{T}L(x_t,y_t) \tag{18}$$

$$\geq TL(x^*,\sum_{t=1}^{T}y_t/T)-\sum_{t=1}^{T}L(x_t,y_t) \tag{19}$$

$$\geq TL(x^*,y^*)-\sum_{t=1}^{T}L(x_t,y_t), \tag{20}$$

where (18) follows the feasibility of x^*, (19) follows the convexity in y, and (20) follows the definition of y^*. Therefore, we have

$$L(x^*,y^*) - \frac{1}{T}\sum_{t=1}^{T} L(x_t,y_t) \le O\left(\frac{\log T}{T}\right). \tag{21}$$

From (16), we know that $y_t = \arg\min_{y\ge 0} L(x_t,y)$. It follow from Lemma 3 that

$$\begin{aligned}
0 &\ge \sum_{t=1}^{T} L(x_t,y_t) - \min_{y\ge 0}\sum_{t=1}^{T} L(x_t,y) \\
&\ge \sum_{t=1}^{T} L(x_t,y_t) - L(x_t,y^*) \qquad (22)\\
&\ge \sum_{t=1}^{T} L(x_t,y_t) - TL(\sum_{t=1}^{T} x_t/T, y^*) \qquad (23)\\
&\ge \sum_{t=1}^{T} L(x_t,y_t) - TL(x^*,y^*), \qquad (24)
\end{aligned}$$

where (22) follows the feasibility of y^*, (23) follows the concavity of L in x, and (24) follows the definition of x^*. Therefore, we have

$$\frac{1}{T}\sum_{t=1}^{T} L(x_t,y_t) - L(x^*,y^*) \le 0. \tag{25}$$

Combining (21) and (25), we can conclude that

$$\left|\frac{1}{T}\sum_{t=1}^{T} L(x_t,y_t) - L(x^*,y^*)\right| \le O\left(\frac{\log T}{T}\right). \quad \square$$

B.2. Proof of Theorem 3

Proof of Theorem 3 Let $x_p = \rho_j(p)z_j$ for all $p\in\mathcal{P}_j, j\in\mathcal{J}$, $\sum_{p\in\mathcal{P}_j}\rho_j(p) = 1$ for all $j\in\mathcal{J}$ and $\rho_j(p)\ge 0$ for all $p\in\mathcal{P}_j, j\in\mathcal{J}$. We define $H(z,y^*) = \sum_{j\in\mathcal{J}} r_j(z_j) - \sum_{e\in\mathcal{E}} c_e \sum_{j\in J}\sum_{p\in\mathcal{P}_j}\delta_e(p)\rho_j(p)z_j + \sum_{e\in\mathcal{E}} y_e^*(k_e - \sum_{j\in J}\sum_{p\in\mathcal{P}_j}\delta_e(p)\rho_j(p)z_j)$. That is, we have $H(z,y^*) = L(x,y^*)$. It is easy to verify that H is μ-strongly concave in z, so we have

$$H(z,y^*) \le H(z^*,y^*) + \nabla_z H(z^*,y^*)^\top(z-z^*) - \frac{\mu}{2}\|z-z^*\|^2.$$

Because z^* is a maximizer, it follows that $\nabla_z H(z^*,y^*)^\top(z-z^*)\le 0$, and hence

$$H(z,y^*) \le H(z^*,y^*) - \frac{\mu}{2}\|z-z^*\|^2.$$

Because $L(x,y^*) = H(z,y^*)$, we have

$$TL(x^*,y^*) - \sum_{t=1}^{T} L(x_t,y^*) \ge \frac{\mu}{2}\sum_{t=1}^{T}\|z_t - z^*\|^2.$$

It follows that

$$\begin{aligned}
TL(x^*,y^*) - \sum_{t=1}^{T} L(x_t,y^*) &= TL(x^*,y^*) - \sum_{t=1}^{T} L(x_t,y_t) + \sum_{t=1}^{T} L(x_t,y_t) - \sum_{t=1}^{T} L(x_t,y^*) \\
&\le O(\log T),
\end{aligned}$$

where the inequality follows from (20) and (24). Therefore,

$$\begin{aligned} O(\log T) &\geq \sum_{t=1}^{T} \|z_t - z^*\|^2 \\ &= T\left[\frac{\sum_{t=1}^{T}\|z_t - z^*\|^2}{T}\right] \\ &\geq T\left[\left\|\sum_{t=1}^{T} z_t/T - z^*\right\|^2\right], \end{aligned}$$

where the last equation follows from Jensen's inequality because $\|\cdot\|^2$ is convex. Therefore, we have

$$\|\bar{z} - z^*\|^2 \leq O\left(\frac{\log T}{T}\right). \quad \square$$

C. Lemmas

Lemma 1. *Let $x^* = \arg\max_{x\in X} f(x)$. When f is β-smooth, under Algorithm 1, we have*

$$f(x^*) - f(x_T) \leq O\left(\frac{1}{T}\right).$$

Proof of Lemma 1 Since f is β-smooth, we have

$$\begin{aligned} f(x_{s+1}) &\geq f(x_s) + \nabla f(x_s)^\top (x_{s+1} - x_s) - \frac{\beta}{2}\|x_{s+1} - x_s\|^2 \\ &= f(x_s) + \nabla f(x_s)^\top (x_s + \gamma_s(w_s - x_s) - x_s) - \frac{\beta}{2}\|x_s + \gamma_s(w_s - x_s) - x_s\|^2 \\ &\geq f(x_s) + \gamma_s \nabla f(x_s)^\top (w_s - x_s) - \frac{\beta\gamma_s^2 R^2}{2} \\ &\geq f(x_s) + \gamma_s \nabla f(x_s)^\top (x^* - x_s) - \frac{\beta\gamma_s^2 R^2}{2} \qquad (26) \\ &\geq f(x_s) + \gamma_s (f(x^*) - f(x_s)) - \frac{\beta\gamma_s^2 R^2}{2}, \qquad (27) \end{aligned}$$

where (26) follows the definition of z_s and (27) holds because f is concave.

$$f(x^*) - f(x_{s+1}) \leq (1-\gamma_s)(f(x^*) - f(x_s)) + \frac{\beta\gamma_s^2 R^2}{2}. \qquad (28)$$

Let $\Delta_s = f(x^*) - f(x_s)$. (28) can be re-written as

$$\Delta_{s+1} \leq (1-\gamma_s)\Delta_s + \gamma_s^2 \frac{\beta R^2}{2}.$$

When $\Delta_s = \frac{2}{s+2}$. We can show by induction that $\Delta_s \leq \frac{2\beta R^2}{s+2}$ for $s = 0, 1, \ldots$. For base case ($s = 0$), we have $\gamma_0 = 1$, and hence, $\Delta_1 \leq \frac{\beta R^2}{2} \leq \beta R^2$. Assume that $\Delta_s \leq \frac{2\beta R^2}{s+2}$ holds for $s \geq 0$. We have

$$\begin{aligned} \Delta_{s+1} &\leq \left(1 - \frac{2}{s+2}\right)\frac{2\beta R^2}{s+2} + \left(\frac{2}{s+2}\right)^2 \frac{\beta R^2}{2} \\ &= \frac{2s\beta R^2}{(s+2)^2} + \frac{2\beta R^2}{(s+2)^2} \\ &= \frac{2\beta R^2}{(s+2)^2}(s+1) \\ &\leq \frac{2\beta R^2}{(s+1)(s+3)}(s+1) \qquad (29) \\ &= \frac{2\beta R^2}{s+3}, \end{aligned}$$

where the inequality (29) holds because $(s+2)^2 \geq (s+1)(s+3)$. $\square$

LEMMA 2 **(Theorem 3.3 in Hazan (2016))**. *For any sequences of μ-strongly convex functions ℓ_t, Online gradient descent (Algorithm 3) with step sizes $\eta_t = \frac{1}{\mu t}$, it follows that*

$$\sum_{t=1}^{T} \ell_t(x_t) - \min_{x\in X} \sum_{t=1}^{T} \ell_t(x) \le \frac{G^2}{2\alpha}(1+\log T),$$

where $\|\nabla f(x)\| \le G$ for all $x \in X$.

Algorithm 5 Best Response

```
Input: $T$
for $t = 0, 1, 2, \dots, T$ do
    Set $x_t \leftarrow \arg\min_{x\in X} \ell_t(x)$
```

LEMMA 3. *For any sequences of convex functions ℓ_t, Best Response (Algorithm 5) achieves*

$$\sum_{t=1}^{T} \ell_t(x_t) - \min_{x\in X} \sum_{t=1}^{T} \ell_t(x) \le 0.$$

Proof of Lemma 3 From the description of Best Response, for $t = 1, \dots, T$, we have $\ell_t(x_t) \le \ell_t(x)$ for all $x \in X$. It follows that

$$\begin{aligned} \sum_{t=1}^{T} \ell_t(x_t) &\le \sum_{t=1}^{T} \ell_t(x), \quad \forall x \in X \\ &\le \min_{x\in X} \sum_{t=1}^{T} \ell_t(x). \quad \square \end{aligned}$$

References

Ahuja RK, Magnanti TL, Orlin JB (1993) Network flows: Theory. *Algorithms, and Applications* 526.

Assad AA (1978) Multicommodity network flows—a survey. *Networks* 8(1):37–91.

Crainic TG (2000) Service network design in freight transportation. *European journal of operational research* 122(2):272–288.

Dijkstra EW (1959) A note on two problems in connexion with graphs. *Numerische mathematik* 1(1):269–271.

Ford LR, Fulkerson DR (1958) A suggested computation for maximal multi-commodity network flows. *Management Science* 5(1):97–101.

Frank M, Wolfe P (1956) An algorithm for quadratic programming. *Naval research logistics quarterly* 3(1-2):95–110.

Hazan E (2016) Introduction to online convex optimization. *Foundations and Trends® in Optimization* 2(3-4):157–325.

Jaggi M (2013) Revisiting frank-wolfe: Projection-free sparse convex optimization. *Proceedings of the 30th International Conference on Machine Learning*, 427–435.

Kennington JL (1978) A survey of linear cost multicommodity network flows. *Operations Research* 26(2):209–236.

Krile S (2004) Application of the minimum cost flow problem in container shipping. *Proceedings. Elmar-2004. 46th International Symposium on Electronics in Marine*, 466–471 (IEEE).

Levitin ES, Polyak BT (1966) Constrained minimization methods. *USSR Computational mathematics and mathematical physics* 6(5):1–50.

Lin CC, Lee SC (2015) Zone pricing for time-definite LTL freight transportation with elastic demand. *Computers & Operations Research* 62:51–60.

Lin CC, Lin DY, Young MM (2009) Price planning for time-definite less-than-truckload freight services. *Transportation Research Part E: Logistics and Transportation Review* 45(4):525–537.

Mitra D, Ramakrishnan K, Wang Q (2001) Combined economic modeling and traffic engineering: Joint optimization of pricing and routing in multi-service networks. *Proc. 17th International Teletraffic Congress*, 73–85.

Ouorou A, Mahey P, Vial JP (2000) A survey of algorithms for convex multicommodity flow problems. *Management science* 46(1):126–147.

Resende MG, Pardalos PM (2008) *Handbook of optimization in telecommunications* (Springer Science & Business Media).

Sharkey TC (2011) Network flow problems with pricing decisions. *Optimization Letters* 5(1):71–83.

Vaidyanathan B, Jha KC, Ahuja RK (2007) Multicommodity network flow approach to the railroad crew-scheduling problem. *IBM Journal of Research and Development* 51(3.4):325–344.

Wieberneit N (2008) Service network design for freight transportation: a review. *OR spectrum* 30(1):77–112.

Zinkevich M (2003) Online convex programming and generalized infinitesimal gradient ascent. *Proceedings of the 20th International Conference on Machine Learning*, 928–936.