Proofs of Statements

EC.1. Supplement to Section 3.1

In this section, we formally prove the reduction of the optimization problem for the class of linear-plus-uplift functions to (6), and then show Propositions 1 and 2.

EC.1.1. Reduction

Here we show that for the class of linear-plus-uplift price functions $p(q_i;\lambda,u_i,\hat{q}_i)=\lambda q_i+u_i\mathbb{1}_{q_i=\hat{q}_i}$, one can assume $\hat{q}_i^*=q_i^*$ without loss of generality, and therefore the optimization problem (5) reduces to (6) for this class. The optimization problem (5) for price function $p(q_i;\lambda,u_i,\hat{q}_i)=\lambda q_i+u_i\mathbb{1}_{q_i=\hat{q}_i}$, $\lambda,u_1,\dots,u_n\geq 0$, is as follows

$$p^*_{\text{uplift}}=\min_{\substack{q_1,\dots,q_n\\ \lambda\geq 0\\ u_1,\dots,u_n\geq 0\\ \hat{q}_1,\dots,\hat{q}_n}} \quad \sum_{i=1}^n(\lambda q_i+u_i\mathbb{1}_{q_i=\hat{q}_i}) \tag{EC.1a}$$

$$\text{s.t.} \quad \sum_{i=1}^n q_i=d \tag{EC.1b}$$

$$\lambda q_i+u_i\mathbb{1}_{q_i=\hat{q}_i}-c_i(q_i)\geq 0, \quad i=1,\dots,n \tag{EC.1c}$$

$$\lambda q_i+u_i\mathbb{1}_{q_i=\hat{q}_i}-c_i(q_i)\geq \max_{q_i'\neq q_i}\ \lambda q_i'+u_i\mathbb{1}_{q_i'=\hat{q}_i}-c_i(q_i'), \quad i=1,\dots,n \tag{EC.1d}$$

The following lemma shows that this optimization problem can be reduced to (6), and the optimal uplifts of (6) are no larger than those of (EC.1).

LEMMA EC.1. *Given any solution* $(\boldsymbol{q}^*,\lambda^*,\boldsymbol{u}^*,\hat{\boldsymbol{q}}^*)$ *to the optimization problem* (EC.1), $(\boldsymbol{q}^*,\lambda^*,\underline{\boldsymbol{u}},\boldsymbol{q}^*)$ *is also a solution, where*

$$\underline{u}_i=\begin{cases} u_i^*, & \text{if } \hat{q}_i^*=q_i^*\\ 0, & \text{o.w.}\end{cases}.$$

Proof of Lemma EC.1. Let us first show the feasibility of $(\boldsymbol{q}^*,\lambda^*,\underline{\boldsymbol{u}},\boldsymbol{q}^*)$. For any i such that $\hat{q}_i^*\neq q_i^*$, we have that

$$\lambda^* q_i^*-c_i(q_i^*)\geq 0$$

$$\lambda^* q_i^* - c_i(q_i^*) \geq \max_{q_i' \neq q_i^*} \ \lambda^* q_i' + u_i^* \mathbb{1}_{q_i' = \hat{q}_i^*} - c_i(q_i') \geq \max_{q_i' \neq q_i^*} \ \lambda^* q_i' - c_i(q_i'),$$

which implies

$$\lambda^* q_i^* + \underline{u}_i^* \mathbb{1}_{q_i^* = q_i^*} - c_i(q_i^*) \geq 0$$

$$\lambda^* q_i^* + \underline{u}_i^* \mathbb{1}_{q_i^* = q_i^*} - c_i(q_i^*) \geq \max_{q_i' \neq q_i^*} \ \lambda^* q_i' + \underline{u}_i^* \mathbb{1}_{q_i' = \hat{q}_i^*} - c_i(q_i'),$$

because $\underline{u}_i^* = 0$. Therefore $(\boldsymbol{q^*}, \lambda^*, \underline{\boldsymbol{u}}, \boldsymbol{q^*})$ is feasible.

The objective value of $(\boldsymbol{q^*}, \lambda^*, \underline{\boldsymbol{u}}, \boldsymbol{q^*})$ is

$$\begin{aligned}\sum_{i=1}^n (\lambda^* q_i^* + \underline{u}_i) &= \sum_{i: \hat{q}_i^* = q_i^*} (\lambda^* q_i^* + u_i^*) + \sum_{i: \hat{q}_i^* \neq q_i^*} \lambda^* q_i^* \\ &= \sum_{i=1}^n (\lambda^* q_i^* + u_i^* \mathbb{1}_{q_i^* = \hat{q}_i^*}),\end{aligned}$$

which is the same as that of $(\boldsymbol{q^*}, \lambda^*, \boldsymbol{u^*}, \boldsymbol{\hat{q}^*})$, and is therefore optimal. □

Based on this lemma, the optimization problem (EC.1) can be reduced to (6).

EC.1.2. Closed-Form Solutions

Proof of Proposition 1. In the optimization problem (6), the order of variables in the minimizations does not matter, and further, for every fixed $q_1, \ldots, q_n$ and λ, the minimization over each u_i can be done separately. Therefore this program can be massaged into the following form

$$p^*_{\text{uplift}} = \min_{q_1, \ldots, q_n} \quad \left(\min_{\lambda \geq 0} \ \sum_{i=1}^n g_i(q_i; \lambda) \right) \tag{EC.2a}$$

$$\text{s.t.} \quad \sum_{i=1}^n q_i = d, \tag{EC.2b}$$

where

$$g_i(q_i; \lambda) = \min_{u_i \geq 0} \quad \lambda q_i + u_i \tag{EC.3a}$$

$$\text{s.t.} \quad \lambda q_i + u_i - c_i(q_i) \geq 0, \tag{EC.3b}$$

$$\lambda q_i + u_i - c_i(q_i) \geq \max_{q_i' \neq q_i} \lambda q_i' - c_i(q_i'). \tag{EC.3c}$$

for all $i = 1, \ldots, n$. Constraints (EC.3b) and (EC.3c) can be expressed as

$$\lambda q_i + u_i \geq c_i(q_i),$$

$$\lambda q_i + u_i \geq c_i(q_i) + \max_{q_i' \neq q_i} \lambda q_i' - c_i(q_i').$$

It follows that

$$g_i(q_i; \lambda) = \lambda q_i + u_i^* = c_i(q_i) + \max \left\{ 0, \max_{q_i' \neq q_i} \lambda q_i' - c_i(q_i') \right\}.$$

which is, of course, a function of λ and q_i. Therefore we have

$$\min_{\lambda \geq 0} \sum_{i=1}^{n} g_i(q_i; \lambda) \;=\; \sum_{i=1}^{n} c_i(q_i)$$

and the minimizers λ^* are all values λ for which $\max_{q_i' \neq q_i} \lambda q_i' - c_i(q_i') \leq 0$, which are exactly the elements of $\Lambda = \{\lambda \geq 0 \mid \lambda q \leq c_i(q),\ \forall q, \forall i\}$ (Figure 1 provides a pictorial description of these values). Finally we have the last minimization, which is

$$\min_{q_1, \ldots, q_n} \quad \sum_{i=1}^{n} c_i(q_i) \tag{EC.4a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} q_i = d \tag{EC.4b}$$

and therefore has $q_i^* = q_i^0\ \forall i$ as its optimizer. We also have $u_i^* = c_i(q_i^*) - \lambda^* q_i^*,\ \forall i$. □

Proof of Proposition 2. The steps of the proof are exactly the same as in the previous one, except that the additional minimizer picks the λ with the smallest total uplift $\sum_{i=1}^{n} u_i(\lambda)$, which corresponds to the largest element of Λ. □

EC.2. Supplement to Section 3.2

In this section, we prove Theorem 1, in two parts. First, we show that there exist finite sets $Q, \mathcal{A}', \mathcal{B}'$ for which Algorithm 1 finds an ϵ-approximate solution, and we quantify the sizes of these sets as a function of ϵ. In the second part, we analyze the running time of Algorithm 1.

EC.2.1. ϵ-Accuracy

Let us first state a simple but useful lemma.

LEMMA EC.2 **(δ-discretization)**. *Given a set $\mathcal{C} \subseteq [\underline{L_1}, \overline{L_1}] \times \cdots \times [\underline{L_k}, \overline{L_k}]$, for any $\delta > 0$, there exists a finite set $\mathcal{C}'$ such that*

$$\forall z \in \mathcal{C},\ \exists z' \in \mathcal{C}'\ \ s.t.\ \|z - z'\|_\infty \leq \delta,$$

and further $\mathcal{C}'$ contains at most V/δ^k points, where $V = \prod_{i=1}^{k}(\overline{L_i} - \underline{L_i})$ is a constant (the volume of the box). $\mathcal{C}'$ is said to be a δ-discretization of $\mathcal{C}$.

Let Q, $\mathcal{A}'$ and $\mathcal{B}'$ denote some δ-discretizations of sets $[0, d]$, $\mathcal{A}$ and $\mathcal{B}$, respectively. In other words, for every $q \in [0, d]$, $\alpha \in \mathcal{A}$, and $\beta \in \mathcal{B}$, there exist $q' \in Q$, $\alpha' \in \mathcal{A}'$, and $\beta' \in \mathcal{B}'$, such that $|q - q'| \leq \delta$, $\|\alpha - \alpha'\|_\infty \leq \delta$, and $\|\beta - \beta'\|_\infty \leq \delta$. We can combine all these inequalities as

$$\|(q, \alpha, \beta) - (q', \alpha', \beta')\|_\infty \leq \delta.$$

On the other hand, given that the cost function $c_i(.)$ for each i is Lipschitz on each continuous piece of its domain, there exists a positive constant K_i such that $|c_i(q) - c_i(q')| \leq K_i|q - q'|$, which implies

$$|c_i(q) - c_i(q')| \leq K_i\delta. \tag{EC.5}$$

Similarly, Lipschitz continuity of $p(.;.)$ implies existence of a positive constant K such that $|p(q, \alpha, \beta) - p(q', \alpha', \beta')| \leq K\|(q, \alpha, \beta) - (q', \alpha', \beta')\|_\infty$, which yields

$$|p(q, \alpha, \beta) - p(q', \alpha', \beta')| \leq K\delta. \tag{EC.6}$$

Using Eqs. (EC.5),(EC.6), we can see that for any solution $q_1^*, \ldots, q_n^*, \alpha^*, \beta_1^*, \ldots, \beta_n^*$ to optimization (5), there exists a point $q_1, \ldots, q_n, \alpha, \beta_1, \ldots, \beta_n$ with $q_1, \ldots, q_n \in Q$, $\alpha \in \mathcal{A}'$ and $\beta \in \mathcal{B}'$, for which constraints (5c) and (5d) are violated at most by $(K + K_i)\delta$ and $(2K + 2K_i)\delta$, respectively, and

the objective is larger than p^* at most by $nK\delta$. As a result, this point will be an ϵ-approximate solution if

$$(K+K_i)\delta \le \epsilon \quad \forall i, \tag{EC.7}$$

$$2(K+K_i)\delta \le \epsilon \quad \forall i, \tag{EC.8}$$

$$nK\delta \le n\epsilon. \tag{EC.9}$$

These constraints altogether enforce an upper bound on the value of δ as

$$\delta \le C\epsilon,$$

for some constant C. Therefore if we pick

$$\delta = \frac{d}{\left\lceil \frac{d}{C\epsilon} \right\rceil}, \tag{EC.10}$$

our algorithm is guaranteed to encounter an ϵ-approximate solution while enumerating the points, and $Q = \{0, \delta, 2\delta, \dots, d\}$ is a valid δ-discretization for $[0, d]$, which has $N_q = \left\lceil \frac{d}{C\epsilon} \right\rceil + 1 = O\left(\frac{1}{\epsilon}\right)$ points. The nice thing about this particular choice of δ is that now d can be written as a sum of n elements in Q (because all the elements, including d, are multiples of δ), which allows us to satisfy the Market Clearing condition exactly. Based on Lemma (EC.2), $\mathcal{A}'$ and $\mathcal{B}'$ contain $N_\alpha = O\left(\frac{1}{\delta^{l_1}}\right) = O\left(\frac{1}{\epsilon^{l_1}}\right)$ and $N_\beta = O\left(\frac{1}{\delta^{l_2}}\right) = O\left(\frac{1}{\epsilon^{l_2}}\right)$ points.

Finally, if there are any discontinuities in the cost or price functions, we can simply add them to our discrete sets Q, $\mathcal{A}'$ and $\mathcal{B}'$, and since there are at most a finite number of them, the sizes of the sets remain in the same order, i.e., $N_q = O\left(\frac{1}{\epsilon}\right)$, $N_\alpha = O\left(\frac{1}{\epsilon^{l_1}}\right)$ and $N_\beta = O\left(\frac{1}{\epsilon^{l_2}}\right)$. Next, we calculate the time complexity of Algorithm 1 running on these discrete sets.

EC.2.2. Run-Time Analysis

In this section, we show that Algorithm 1 has a time complexity of $O\left(n(\frac{1}{\epsilon})^{l_1+l_2+2}\right)$. For every fixed α, we have the following computations

1. The leaves: We need to compute $g_i(q;\alpha)$ for every i and every $q \in Q$. Computing each $g_i(q;\alpha)$ (i.e. for fixed i, q, α) takes $O(N_\beta N_q)$. The reason for that is we have to search over all $\beta_i \in B'$, and for each one there are $N_q + 1$ constraints to check. More explicitly, we need to (a) check $O(N_\beta N_q)$ constraints, (b) compute N_β objectives, and (c) find the minimum among those N_β values. All these steps together take $O(N_\beta N_q)$, and repeating for every i and q makes it $O(nN_\beta N_q^2)$.
2. The intermediate nodes: In each new level, there are at most half as many (+1) nodes as in the previous level. For each node i in this level, we need to compute $g_i(q;\alpha)$ for every $q \in Q$. For every fixed q, there are $O(N_q)$ possible pairs of (q_j, q_k) that add up to q, and therefore we need to (a) sum $O(N_q)$ pairs of objective values, and (b) find the minimum among them, which take $O(N_q)$. Hence, the computation for each node takes $O(N_q^2)$. There are $O(\frac{n}{2} + \frac{n}{4} + \cdots + 2) = O(n)$ intermediate nodes in total, and therefore the total complexity of this part is $O(nN_q^2)$.
3. The root: Finally at the root, we need to compute $g_{\text{root}}(d;\alpha)$. There are N_q possible pairs of (q_j, q_k) that add up to d. Therefore, we need to compute N_q sums, and find the minimum among the resulting N_q values, which takes $O(N_q)$.

Putting the pieces together, the computation for all values of α takes $N_\alpha \times \big(O(nN_\beta N_q^2) + O(nN_q^2) + O(N_q)\big)$, which in turn is $O(nN_\alpha N_\beta N_q^2)$. Finally, finding the minimum among the N_α values simply takes $O(N_\alpha)$.

The backward procedure, which finds the quantities q_i and the parameters β_i, takes just $O(n)$, since it is just a substitution for every node. As a result, the total running time is $O(nN_\alpha N_\beta N_q^2)$, which based on the first part (Section EC.2.1) is $O\left(n(\frac{1}{\epsilon})^{l_1+l_2+2}\right)$. □

EC.2.3. Remark on the ϵ-Approximation

As mentioned at the end of Section 3.2, if one requires the total payment in Definition 1 to be at most ϵ (rather than $n\epsilon$) away from the optimal p^*, the running time of our algorithm will still be polynomial in both n and $1/\epsilon$, i.e., $O\left(n^3(\frac{1}{\epsilon})^{l_1+l_2+2}\right)$. To see that, notice in this case (EC.7) and

(EC.8) remain the same, and (EC.9) changes to $nK\delta \leq \epsilon$. Therefore, the upper bound enforced by the constraints will be $\delta \leq \frac{C\epsilon}{n}$, for some constant C. In this case, our choice of δ would be $\delta = \frac{d}{\lceil \frac{dn}{C\epsilon} \rceil}$, and hence $N_q = O\left(\frac{n}{\epsilon}\right)$. N_α and N_β remain the same as before. The running time is $O(nN_\alpha N_\beta N_q^2)$, as computed previously, which in this case would be $O\left(n^3(\frac{1}{\epsilon})^{l_1+l_2+2}\right)$.

EC.3. Supplement to Section 4

In this section, we first show the transformation of the problem on a tree to one on a binary tree, and then prove Theorem 2.

EC.3.1. Transformation into Binary Tree

LEMMA EC.3. *Given any tree with n nodes (suppliers), there exists a binary tree with additional nodes which has the same solution $(q_i^*, \ldots, q_n^*, \alpha^*, \beta_1, \ldots, \beta_n)$ for those nodes as the original network. The binary tree has $O(n)$ nodes.*

Take any node i that has $k_i > 2$ children. For any two children introduce a dummy parent node. For any two dummy parent nodes introduce a new level of dummy parent nodes. Continue this process until there are 2 or less nodes in the uppermost layer, and then connect them to node i (See Fig. EC.1). The capacities of the lines immediately connected to the children are the same as those in the original graph. The capacities of the new lines are infinite.

The total number of introduced dummy nodes by this procedure is

$$O(\frac{k_i}{2} + \frac{k_i}{4} + \cdots + 2) = O(k_i).$$

Since there are $1 + k_1 + k_2 + \cdots + k_n = n$ nodes in total in the original tree, the number of introduced additional nodes is $O(k_1 + \cdots + k_n) = O(n)$. Therefore the total number of nodes in the new (binary) tree is $O(n)$. □

EC.3.2. Proof of Theorem 2

Most of the proof is similar to the one presented in Section EC.2. For this reason, we only highlight the main points. The proof consists of ϵ-accuracy and run-time, as before.

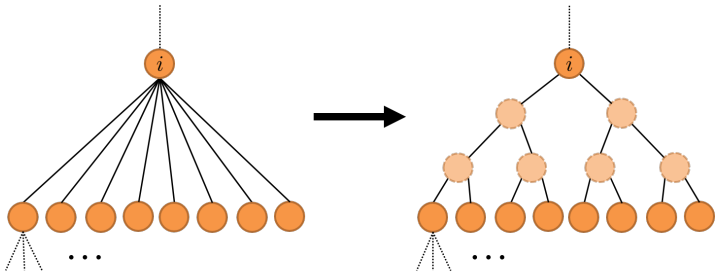

Figure EC.1 The transformation of an arbitrary-degree tree to a binary tree

EC.3.2.1. ϵ-Accuracy

Let $Q_1,\dots,Q_n,F_1,\dots,F_n,\mathcal{A}',\mathcal{B}'$ denote some δ-discretizations of sets $[0,d_1+\overline{f_{\mathrm{ch}_1(1)}}+\overline{f_{\mathrm{ch}_2(1)}}-\underline{f_1}],\dots,[0,d_n+\overline{f_{\mathrm{ch}_1(\mathrm{n})}}+\overline{f_{\mathrm{ch}_2(\mathrm{n})}}-\underline{f_n}]$, $[\underline{f_1},\overline{f_1}],\dots,[\underline{f_n},\overline{f_n}]$, $\mathcal{A}$, $\mathcal{B}$, respectively. Note that if any line capacities are infinite, the intervals can be replaced by $[0,\sum_{i=1}^n d_i]$ instead. Similar as in Section EC.2, the constraints enforce an upper bound on the value of δ as $\delta\le C\epsilon$, for some constant C. Based on Lemma (EC.2), the sizes of the sets will be $N_{q_i}=O\left(\frac{1}{\epsilon}\right)\ \forall i$, $N_{f_i}=O\left(\frac{1}{\epsilon}\right)\ \forall i$, $N_\alpha=O\left(\frac{1}{\epsilon^{l_1}}\right)$ and $N_\beta=O\left(\frac{1}{\epsilon^{l_2}}\right)$

EC.3.2.2. Run-Time Analysis

For every fixed α, the run-time of the required computations is as follows.

1. The time complexity of computing $g_i(q_i;\alpha)$ for each node i and each fixed value of q_i is $O(N_\beta N_{q_i})$. Therefore, computing it for all nodes and all values takes $O(nN_\beta N_q^2)$.
2. Computing $h_i(f_i;\alpha)$ for each node i and each fixed value of f_i takes $O(N_f^2)$, because there are $O(N_f)\times O(N_f)$ pairs of values for $(f_{\mathrm{ch}_1(i)},f_{\mathrm{ch}_2(i)})$ (q_i is automatically determined as the closest point in Q_i to $d_i+f_{\mathrm{ch}_1(i)}+f_{\mathrm{ch}_2(i)}-f_i$). Therefore, its overall computation for all nodes and all values takes $O(nN_f^3)$.

As a result, the overall computation takes $N_\alpha\times\left(O\left(nN_\beta N_q^2\right)+O\left(nN_f^3\right)\right)$, which is $O\left(n(\frac{1}{\epsilon})^{l_1+l_2+2}\right)+O\left(n(\frac{1}{\epsilon})^{l_1+3}\right)$, or equivalently $O\left(n(\frac{1}{\epsilon})^{l_1+\max\{l_2,1\}+2}\right)$. □