# Optimization Course Project II:

# Online Resource Allocation in Auction and Price-Posting Markets

## 1 Introduction

The online resource allocation problem aims to allocate limited resources to heterogeneous agents that appear sequentially. This problem can be applied to many topics, such as revenue management, healthcare, and fairness. This project focuses on the online allocation problem in the auction and price-posting markets, as we discussed in class.

## 2 Online Auction Market

We first consider the online auction market. Here, decision-maker has $m$ different resources with inventory $\{b_i\}_{i=1}^n$. There are $n$ agents in this market have resource requests $\{\mathbf{a}_j = \{a_{ij}\}_{i=1}^m\}_{j=1}^n$ and linear reward functions $\{u_j(x_j)\}_{j=1}^n = \pi_j x_j$, where $x_j$ denotes the fraction that the $j$-th bidder is satisfied. For example, $x_j = 1$ and 0 represent that the $j$-th bidder is totally accepted or rejected, respectively. Then, the decision-maker's goal is to maximize the total revenue by allocating resources. The corresponding optimization problem can be written as follows:

$$\begin{array}{ll} \text{maximize}_{\mathbf{x}} & \sum_{j=1}^n u_j(x_j) \\ \text{s.t.} & \sum_{j=1}^n a_{ij} x_j \le b_i, \ \forall\ i = 1, 2, ..., m \\ & 0 \le x_j \le 1, \ \forall\ j = 1, ..., n. \end{array} \tag{1}$$

For more details, please see [3] and references therein.

The classical offline LP algorithm would compute the full optimal solution $\mathbf{x}$ in one go, while the online algorithm would compute the solution sequentially $x_1$, then $x_2, \ldots$ Specifically, when computing the decision variables $x_1$ to $x_k$, we don't consider information associated with $x_{k+1}$ and beyond. In this course project, you are asked to study and explore some theories of this online allocation problem and perform computational observations on some simulated or real data.

## 2.1 Possible Approaches

Below are suggested approaches for solving the problem.

**Approach 1.1:** The one-time learning online algorithm described in [3] is as follows: suppose there is a reliable estimate that there will be a total of $n$ bidders in the market; then we wait for the first $k$ bidders to arrive, where $k$ can be $50, 100, 200, ...$, and solve the resulting partial linear program:

$$\begin{array}{lll} \text{(SLPM):} & \text{maximize}_{x_1,...,x_k} & \sum_{j=1}^{k} \pi_j x_j \\ & \text{s.t.} & \sum_{j=1}^{k} a_{ij} x_j \leq \frac{k}{n} b_i, \ \forall\ i = 1, 2, ..., m, \\ & & 0 \leq x_j \leq 1, \ \forall\ j = 1, ..., k. \end{array}$$

and then use the *dual prices*, say $\bar{\mathbf{y}}^k$, of the (partial) LP for future online decisions:

$$x_j = 1, \text{ if } \pi_j > \mathbf{a}_j^T \bar{\mathbf{y}}^k \text{ and there are remaining goods left;} \quad \text{and } x_j = 0, \text{ otherwise.}$$

The interpretation is that we allow the allocation to bidder $j$ if their bid is higher than the value, calculated after $k$ bids, of the goods they require.

**Approach 1.2:** In addition, [3] also introduce a dynamical updating algorithm. In this algorithm, the dual prices will be recomputed at time points $k = 50, 100, 200, 400, 800...$ and used to make decisions for the immediate subsequent period.

**Approach 2.1:** The online algorithm described above does not use information on how much good inventory remains for allocation in the decision process. One approach is to consider an offline model as

$$\begin{array}{ll} \text{maximize}_{\mathbf{x},\mathbf{s}} & \sum_j \pi_j x_j + u(\mathbf{s}) \\ \text{s.t.} & \sum_j a_{ij} x_j + s_i = b_i, \ \forall\ i = 1, 2, ..., m, \\ & 0 \leq x_j \leq 1, \ \forall\ j = 1, ..., n, \\ & s_i \geq 0, \forall\ i = 1, ..., m. \end{array} \tag{2}$$

where $u(\mathbf{s}) = u(s_1, \ldots, s_m)$ is increasing and strictly concave and it gradient entry $\frac{\partial u(.)}{\partial s_i}_{s_i=0}$ is sufficiently large for all $i$. Typical choices of $u(\mathbf{s})$ are

$$u(\mathbf{s}) = \frac{w}{m} \sum_i \log s_i$$

or

$$u(\mathbf{s}) = \frac{w}{m} \sum_i (1 - e^{-a \cdot s_i})$$

for some parameters $w(> 0)$ and $a(> 0)$.

This approach solves the revealed partial convex program after the first $k$ bidders arrived:

$$\begin{array}{lll} \text{(SCPM):} & \text{maximize}_{x_1,...,x_k} & \sum_{j=1}^{k} \pi_i x_i + u(\mathbf{s}) \\ & \text{s.t.} & \sum_{j=1}^{k} a_{ik} x_k + s_i \leq \frac{k}{n} b_i, \ \forall\ i = 1, 2, ..., m, \\ & & 0 \leq x_j \leq 1, \ \forall\ j = 1, ..., k. \end{array}$$

and then use the *optimal Lagrangian multipliers/prices*, say $\bar{\mathbf{y}}^k$, of the partial convex program for future online decisions:

$$x_j = 1, \text{ if } \pi_j > \mathbf{a}_j^T \bar{\mathbf{y}}^k \text{ and there are remaining goods left;} \quad \text{and } x_j = 0, \text{ otherwise.}$$

**Approach 2.2:** Similar to the dynamic updating algorithm in Approach 1.2, the algorithm in Approach 2.1 also has a dynamic updating version. Specifically, one can dynamically update the *optimal Lagrangian multipliers/prices* at time points $k = 50, 100, 200, 400, 800...$ and use them to make decisions for the immediate subsequent period.

**Approach 3** Another online approach that uses the information of inventory (See [6]) is the Action-history-dependent Learning Algorithm, which means this algorithm uses historic actions to update the dual price, i.e., at each time point $k \geq 2$, decide the value of $x_k$:

$$x_k = 1, \text{ if } \pi_k > \mathbf{a}_k^T \bar{\mathbf{y}}^{k-1} \text{ and there are remaining goods left;} \quad \text{and } x_k = 0, \text{ otherwise,}$$

update remaining resources:

$$b_i^{(k)} = b_i^{(k-1)} - a_{ik} x_k \text{ for i = 1,2,...,m,}$$

and, then update the dual price by solving the linear programming:

$$\begin{array}{ll} \text{maximize}_{x_1,...,x_k} & \sum_{j=1}^k \pi_j x_j \\ \text{s.t.} & \sum_{j=1}^k a_{ij} x_j \leq \frac{k}{n-k} b_i^{(k)}, \ \forall\ i = 1, 2, ..., m, \\ & 0 \leq x_j \leq 1, \ \forall\ j = 1, ..., k. \end{array} \tag{3}$$

**Approach 4:** Although both dynamic online algorithms above can provide good results, one drawback is that one has to solve a large-scale LP to update the dual price. When $n$ is large, the computation is time- and memory-consuming. A way to fix it is to utilize the idea of Stochastic Gradient Descent. Specifically, in [6], note that if $(\pi_j, \mathbf{a}_j) \sim (\pi, \mathbf{a}),\ j = 1, 2, ..., n$ is a sequence of i.i.d. random vectors, the problem (1) is closely related to

$$\begin{array}{ll} \text{minimize}_{\bar{\mathbf{y}}} & \mathbf{d}^T \bar{\mathbf{y}} + \mathbb{E}\left(\pi - \mathbf{a}^T \bar{\mathbf{y}}\right)^+, \\ \text{s.t.} & \bar{\mathbf{y}} \geq 0, \end{array} \tag{4}$$

where $\mathbf{d} = \mathbf{b}/n$ and $(\cdot)^+ = \max\{\cdot, 0\}$. Under some mild conditions, the objective in (4) is differentiable and the derivative is

$$\mathbf{f}(\bar{\mathbf{y}}) = \mathbb{E}\left(\mathbf{d} - \mathbf{a}\mathbb{I}_{\{\pi > \mathbf{a}^T \bar{\mathbf{y}}\}}\right).$$

Assume at each time step $k$, $\mathbb{E}(\mathbf{f}(\pi, \mathbf{a}))$ can be approximated by $\mathbf{f}(\pi_k, \mathbf{a}_k)$. Then, one can apply stochastic gradient descent to $\mathbf{f}(\cdot)$ to update the price and use them to make decisions for the immediate subsequent period.

One may apply the same idea to solve the dual of (2) to update prices.

### 2.2 Project Goal:

You may explore this problem by generating a large-scale online allocation problem with $n = 10000$ agents, $m = 10$ resources, and $b_i = 1,000$ for all $i$. One way to generate a sequence of random bids, $k = 1, 2, ...$, is as follows: first fix a ground truth price vector $\bar{\mathbf{p}} > 0$; generate a vector $\mathbf{a}_k$ whose each entry is either zero or one at random, then let $\pi_k = \bar{\mathbf{p}}^T \mathbf{a}_k + normrnd(0, 0.2)$ where $normrnd(0, 0.2)$ represents the Gauss random variable with zero mean and variance 0.2 in Matlab.

The key questions are: what are the best approaches to approximately solve the online resource allocation problem in the auction market? What are the best approaches to find the optimal dual price of (4)? Will a better estimation of the optimal solution of (4) lead to a better allocation? What are the differences among different methods? The comparison can include aspects such as algorithm design, theoretical analysis, computation time, and the approximation error of different algorithms.

## 3 Online Price-Posting Market

In contrast to the online auction market, where the decision-maker decides the allocation to each agent, in the price-posing market, the decision-maker only posts a price for each resource, and agents decide the final allocations based on their budgets and the given price.

In this project, we consider the fisher market. specifically, there are $m$ goods in the market and each good $i$ has a fixed amount $\bar{s}_i (> 0)$ available. There are $n$ buyers in the market where each buyer, say buyer $j \in \{1, ..., n\}$, is equipped with a fixed budget $w_j (> 0)$ and independently solves a linear utility maximization problem

$$\max_{\mathbf{x}_j} \quad u_j(\mathbf{x}_j) = \sum_{i=1}^{m} u_{ij} x_{ij} \quad \text{s.t. } \mathbf{p}^T \mathbf{x}_i \le w_j, \ \mathbf{x}_j \ge 0. \tag{5}$$

Here $u_{ij} \ge 0$ is the utility of buyer $j$ on good $i$, and decision variable $x_{ij}$ is the amount of good $i$ purchased by buyer $j$, and $\mathbf{p} \in R^m$ is a given market price vector. The equilibrium prices are the prices to clear the market.

According to Eisenberg and Gale, one can find the equilibrium price based on the following program if the information of all customers is collected:

$$\begin{aligned} \max_{\mathbf{x}_j} \quad & \sum_{j=1}^{n} w_j \log \left( \sum_{i=1}^{m} u_{ij} x_{ij} \right) \\ \text{s.t.} \quad & \sum_{j=1}^{n} \mathbf{x}_i \le \bar{\mathbf{s}}, \ \mathbf{x}_j \ge 0, \end{aligned} \tag{6}$$

where $\bar{\mathbf{s}} = (\bar{s}_1, \ldots, \bar{s}_m)^\top$.

This project considers an online fisher market where customers arrive sequentially. At each time, the decision-maker can only post a price, and customers will trade by solving (5). The goal of the decision-maker is to maximize the total social welfare and clear the market. That is, to solve (6) approximately in an online fashion.

### 3.1 Project Goal

To explore this question, you may generate 10000 customers as in [8], and generalize some algorithms in Section 2.1 to this online fisher-market problem. Will those algorithms still work? You can assume that their utility functions and budgets will be revealed upon arrival or that they are not always available to the decision-maker.

## References


[1] Shipra Agrawal and Nikhil R. Devanur. Fast Algorithms for Online Stochastic Convex Programming http://arxiv.org/abs/1410.7596, SODA2015

[2] S. Agrawal, E. Delage, M. Peters, Z. Wang and Y. Ye. A Unified Framework for Dynamic Prediction Market Design. Operations Research, 59:3 (2011) 550-568.

[3] S. Agrawal, Z. Wang and Y. Ye. A Dynamic Near-Optimal Algorithm for Online Linear Programming. http://arxiv.org/abs/0911.2974; to appear in Operations Research.

[4] Anupam Gupta and Marco Molinaro How the Experts Algorithm Can Help Solve LPs Online. https://arxiv.org/abs/1407.5298, 2014.

[5] Thomas Kesselheim, Klaus Radke, Andreas Tönnis, Berthold Vöcking. Primal Beats Dual on Online Packing LPs in the Random-Order Model. https://arxiv.org/abs/1311.2578, STOC 2014.

[6] X. Li and Y. Ye. Online Linear Programming: Dual Convergence, New Algorithms, and Regret Bounds. https://arxiv.org/abs/1909.05499

[7] M. Peters, A. M-C. So and Y. Ye. Pari-mutuel Markets: Mechanisms and Performance. *The 3rd International Workshop On Internet And Network Economics*, 2007.

[8] Jalota, Devansh, and Yinyu Ye. Stochastic Online Fisher Markets: Static Pricing Limits and Adaptive Enhancements.