CME 307: Optimization Final (Project II)

Students: Casey Fleeter & Laura Lyman
Due on Monday 03/20/17

Background. Consider m distinct states that are mutually exclusive such that exactly one of them will be true with time. Define a *share contract* of states to be a paper agreement ensuring that a bet is worth \$1 if the bet includes the winning state and is worth \$0 otherwise. Players can bet on one or a combination of states. We refer to bidders as *orders*, so the jth bidder is sometimes called the jth order. There are n bidders/orders total.

Let $a_j \in \mathbb{R}^m$ denote the combination betting vector $(a_{1j}; \cdots ; a_{mj})$ chosen by the jth bidder, where

$$a_{kj} = \begin{cases} 1 & \text{if state } k \text{ is included in order } j\text{'s betting vector} \\ 0 & \text{else.} \end{cases}$$

Let $q_j \in \mathbb{R}_{>0}$ be the maximum number of shares that the jth bidder requested, and let π_j be the price given to the jth bidder per share. Let x_j be the number of shares actually awarded to the jth order, meaning the jth bidder will pay $\pi_j x_j$. Consequently, the market maker collects a total of

$$\sum_{j=1}^{n} \pi_j x_j = \pi^\intercal x.$$

Now suppose state k is the winning state. Then each a_j such that $a_{kj} = 1$ will be awarded \$1 per share, and the remaining orders win nothing. So the market maker needs to pay back the total amount

$$\sum_{j=1}^{n} a_{kj} x_j = a_k \vec{x}$$

where a_k in this case is the kth row of matrix A. The goal of the market maker is then to choose $\vec{x} \in \mathbb{R}^n$ such that his/her profit is maximized. That is, the market maker needs to pick how many shares x_j to award to order j for all $j \in \{1, \dots, m\}$. Now let $z \in \mathbb{R}_{>0}$ be the worst-case cost for the market maker, so the total worst-case profit is $\pi^\intercal x - z$. So $z = \max_i \vec{a_i}\vec{x}$. Then we want to maximize $\pi^\intercal x - z$ (subject to certain constraints) so that even in the worst-case the market maker has as much profit as possible.

We now introduce a slack variable $s \in \mathbb{R}^m$. Let $u(s)$ be a real-valued function for the market on possible slack shares. Let $A\vec{x} - \vec{e} \cdot z$ and $x \leq \vec{q}$ be associated with dual/Lagrange variables $\vec{p} \in \mathbb{R}^m_{\geq 0}$ and $s \in \mathbb{R}_{\geq 0}$. In linear programs, $\vec{p}$ is interpreted as the state implicit prices (p_k is price for the kth state) and $\vec{s}$ is seen as the order implicit prices (that is, s_j is how much profit per share the marker makes can yield from the jth order). Note that if $u(\cdot)$ is strictly concave, then the state price vector $\vec{p}$ is *unique.*

Suppose the kth order just arrived. Then in the *online* auction market, the problems at hand are:

$$\text{(SCPM)} \quad \begin{array}{ll} \max_{x_k, s} & \pi_k - z + u(s) \\ \text{subject to} & a_{ik} x_k - z + s_i = -\sum_{j=1}^{k-1} a_{ij} \overline{x}_j \quad \forall\, i \in \{1, \dots m\} \\ & 0 \leq x_k \leq q_k \end{array}$$

$$\begin{array}{llll}\text{(SLPM)} & \max_{x_1,\ldots,x_k} & \sum_{j=1}^{k} \pi_j x_j - z & \\ & \text{subject to} & \sum_{j=1}^{k} a_{ij} x_j - z \le 0 & \forall\ i \in \{1,\ldots m\} \\ & & 0 \le x_j \le q_j & \forall\ j \in \{1,\ldots,k\}\end{array}$$

where $\overline{x}_j$ for $j \in \{1,\ldots,k-1\}$ is the vector of decisions made at previous steps (that is, the vector of shares awarded to previous bidders).

First we implement both SCPM and SLPM for online prediction auctions as described in Lecture Note # 9. For the SCPM, consider the utility functions

$$u_1(s) = \frac{w}{m}\sum_{i=1}^{m}\log(s_i) \quad \text{and} \quad u_2(s) = \frac{w}{m}\sum_{i=1}^{m}(1-e^{-s_i})$$

for some fixed positive constant $w \in [10,^{-4} 1]$. All online and offline auctions are run using simulated bidding data with $m = 10$ and some $p_{\text{true}} > 0$ as a grand truth price vector.

Q0 (Numerical): What are the performances of SCPM and SLPM both online and offline? Does w make a difference? Does $\vec{p} \to p_{\text{true}}$, where $\vec{p}$ is the state price vector generated from the online auction model and $p_{\text{true}} > 0$ is the grand truth vector? Explain your observations and findings.

Solution: As suggested, a sequence of random bids for $k \in \{1,\ldots,10,000\}$ is produced by: (1) generating a vector $\vec{a}_k$ whose entries are either 0 or 1 at random, (2) choosing a random scalar $q_k \in [10, 20]$, and (3) letting $\pi_k = \overline{p}^T a_k +$ `randn`(0,0.2), where `randn`(0,0.2) is a Gaussian random variable with mean 0 and variance 0.2.

Below we plot the error (with $w = 0.7601$) from implementing Online SLPM and Online SCPM for both utility functions u_1 and u_2. Error is defined as $\|p_{\text{true}} - p\|_2$ where p is the state price vector yielded from each method. The lefthand plot uses u_1 and the righthand plot uses u_2; both have minimal variance in $\vec{\pi}$ ($\sigma^2 = 0.001$). Noise will be added in later examples.

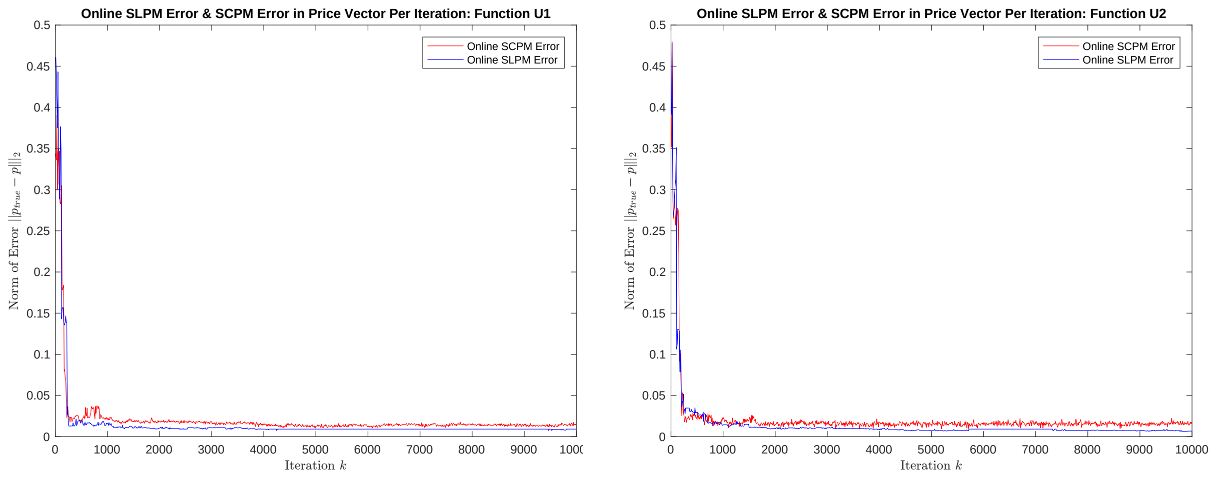


Figure 1: Error in state price vector for $w = 0.7601$ and utility functions u_1 (left) and u_2 (right) over 10,000 iterations via Online SLPM and Online SCPM with $\sigma^2 = 0.001$.

Observe that $p \to p_{\text{true}}$ in Online SCPM and Online SLPM for both utility functions over $n = 10,000$ iterations when there is little/no variance. SLPM converges slightly faster over the iterations, though the difference is fairly minimal. The specific value of w did not seem to make a dramatic difference. Below we show two plots with the same p_{true} and $\sigma^2 = 0.001$ but with a significantly smaller w value ($w = 0.001$ instead of 0.7601).

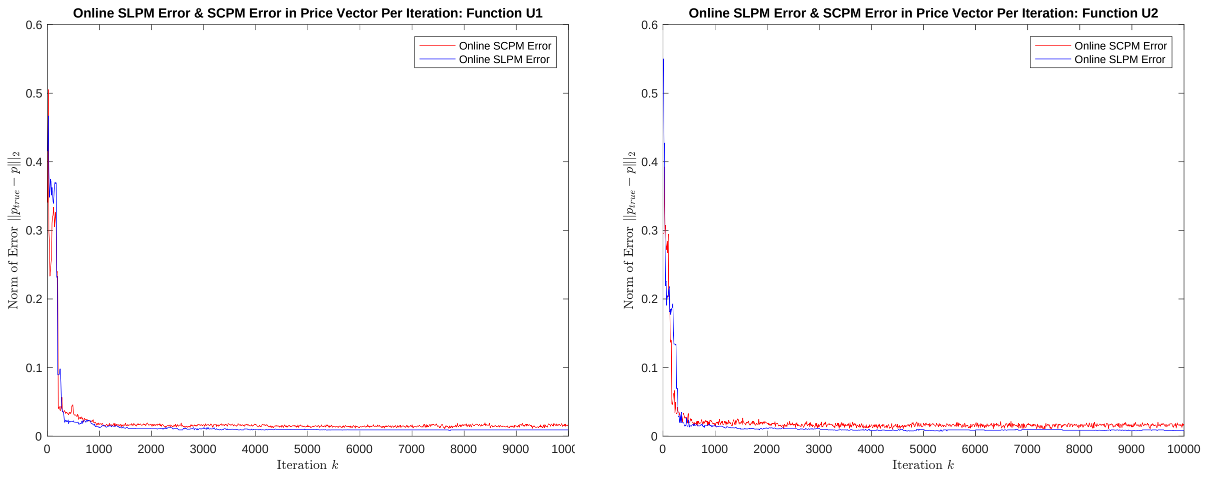


Figure 2: Error in state price vector for $w = 0.001$ and utility functions u_1 (left) and u_2 (right) over 10,000 iterations via Online SLPM and Online SCPM with $\sigma^2 = 0.001$.

As depicted, the convergence for SCPM is nearly identical in this case to its convergence when there is a larger coefficient w. However, once noise is introduced, Online SLPM continues to have $p \to p_{\text{true}}$ while Online SCPM loses convergence to the correct solution. Below are results over 10,000 bids for $w = 0.7601$ and $\sigma^2 = 0.2$ variance.

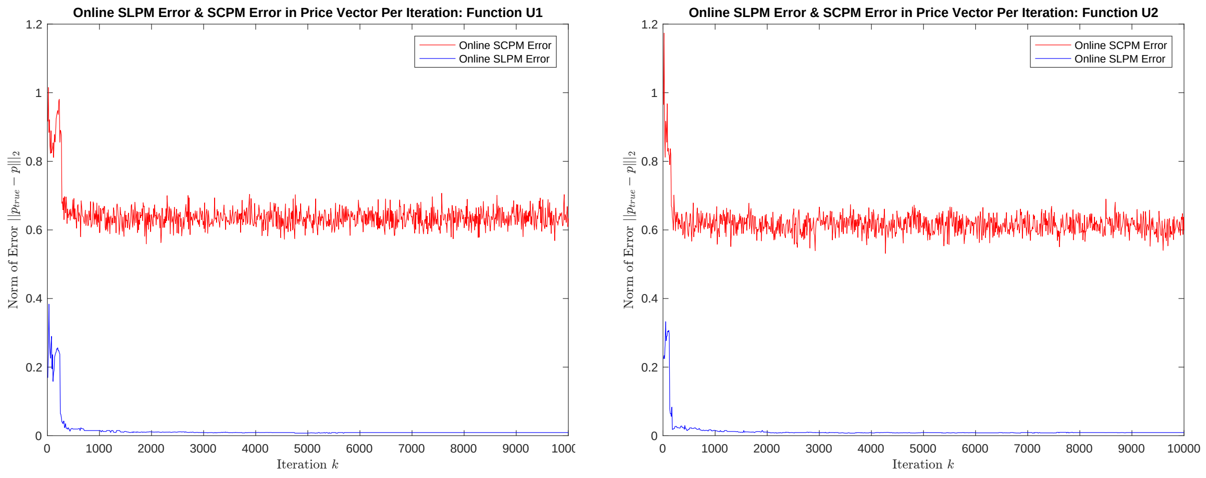
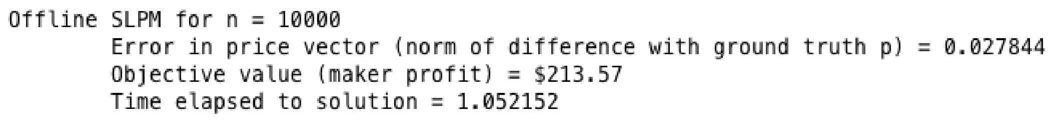


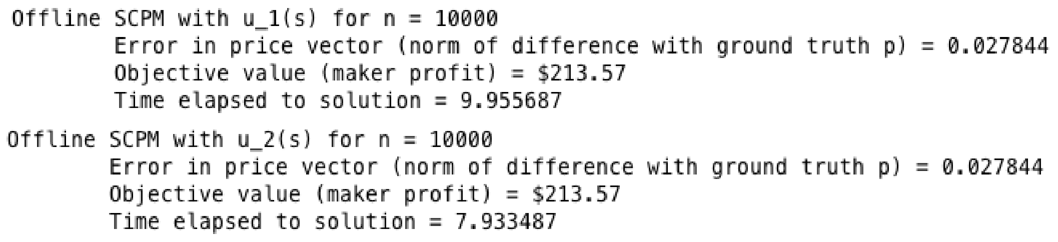
Figure 3: Noise added. Error in state price vector for $w = 0.7601$ and utility functions u_1 (left) and u_2 (right) over 10,000 iterations via Online SLPM and Online SCPM with $\sigma^2 = 0.2$.

In the offline case, SLPM very accurately approximates p_{true} within 10,000 iterations, even with noise introduced ($\sigma^2 = 0.2, w = 0.7601$ below).



```
Offline SLPM for n = 10000
        Error in price vector (norm of difference with ground truth p) = 0.027844
        Objective value (maker profit) = $213.57
        Time elapsed to solution = 1.052152
```

Furthermore, Offline SCPM also converges to p_{true} for $n = 10,000$ ($\sigma^2 = 0.2, w = 0.7601$) for both the logarithmic and exponential utility functions, unlike in the online version.



```
Offline SCPM with u_1(s) for n = 10000
        Error in price vector (norm of difference with ground truth p) = 0.027844
        Objective value (maker profit) = $213.57
        Time elapsed to solution = 9.955687
Offline SCPM with u_2(s) for n = 10000
        Error in price vector (norm of difference with ground truth p) = 0.027844
        Objective value (maker profit) = $213.57
        Time elapsed to solution = 7.933487
```

When w was forced to be small in the offline case ($w = 0.001$), the error and objective values were identical to those corresponding to a higher w value. So similar to the online case, varying w did not seem to affect this convergence given so many bidders, even though w theoretically could modulate the degree to which the market-maker is risk-averse (as described in Q1 below).

Q1 (Theoretical): Dual prices may not be unique when solving the model. In order to get a unique pice, people sometimes add another term to the objective function when solving the model; one such model called CPCAM is defined as follows:

$$\begin{array}{llll}
\text{(CPCAM)} & \max_{x_k,s} & \sum_{j=1}^{n} \pi_j x_j + u(s) & \\
 & \text{subject to} & \sum_{j=1}^{n} a_{ij} x_j + s_i = b_i & \forall\, i \in \{1, \dots m\} \\
 & & 0 \le x_j \le 1 & \forall\, j \in \{1, \dots, n\} \\
 & & s_i \ge 0 & \forall\, i \in \{1, \dots, m\}
\end{array}$$

where $u(\vec{s})$ is increasing and strictly concave. Write down the first-order KKT conditions for optimality. Are they sufficient? Argue why the problem will have a unique price (that is, the Lagrange multipliers on the m equality constraints of the problem). Assume that $\frac{\partial u(\cdot)}{\partial s_i}\,|_{s_i=0} \ge 1$ for all $i \in \{1, \dots m\}$. How would you interpret $u(\vec{s})$?

Solution. First observe that the Lagrangian function is (when ignoring the multiplier for the $\vec{s} \ge 0$ constraint)

$$L(x, s, \lambda_i, \gamma_j, \gamma'_j) = \pi^\intercal x + u(s) - \sum_{i=1}^{m} \lambda_i \big(\sum_{j=1}^{n} a_{ij} x_j + s_i - b_i \big) + \sum_{j=1}^{n} \gamma_j (x_j - 1) - \sum_{j=1}^{n} \gamma'_j x_j j$$

where $\lambda_i, \gamma_j, \gamma'_j, \ge 0$ are Lagrange multipliers encoding the constraints $\sum_{j=1}^{n} a_{ij} x_j + s_i = b_i, \vec{x} \le 1$, and $\vec{x} \ge 0$ respectively. Compute that

$$\frac{\partial}{\partial x_j} L(x, s) = \pi_j - \sum_{i=1}^{m} \lambda_i a_{ij} + \gamma_j - \gamma'_j \qquad \text{for all } j = 1, \dots, n$$

$$\frac{\partial}{\partial s_i} L(x, s) = \frac{\partial}{\partial s_i} u(s) - \lambda_i = 0 \qquad \text{for all } i = 1, \dots, m.$$

Then the KKT conditions enforce that (adding the complementary conditions)

$$\boxed{\begin{aligned} \pi_j - \sum_{i=1}^{m} \lambda_i a_{ij} + \gamma_j - \gamma_j' &= 0 \quad && (\vec{s} \geq 0) \\ \frac{\partial}{\partial s_i} u(s) &= \lambda_i && \text{for } i \in \{1, \dots, m\} \\ \lambda_i \big(\sum_{j=1}^{n} a_{ij} x_j + s_i - b_i \big) &= 0 && \text{for } i \in \{1, \dots, m\} \\ \gamma_j (x_j - 1) &= 0 && \text{for } j \in \{1, \dots, n\} \\ \gamma_j' x_j &= 0 && \text{for } j \in \{1, \dots, n\} \\ \lambda_i, \gamma_j, \gamma_j' &\geq 0 && \text{for all } i, j \\ \sum_i \lambda_i &= 1 && \text{prices sum to 1.} \end{aligned}}$$

Since $-u(s)$ is strictly convex and $-\sum_{j=1}^{n} \pi_j x_j$ is linear, their sum (the negative of the objective function) is a convex function as well. By Lecture Notes # 6, the first-order KKT condition are *sufficient* since the Hessian of the Lagrangian function is negative semidefinite from the the objective function being convex. So a first-order KKT point will also be a 2nd-order KKT point.

Each multiplier λ_i (corresponding to the state prices) has the constraint $\lambda_i = \frac{\partial u(s)}{\partial s_i}$. Since $u(s)$ is strictly concave, it will have a unique maximum solution s^* and thus so will $\frac{\partial u(s^*)}{\partial s_i}$. Therefore, the λ_i are uniquely determined, meaning the problem has a unique price.

We interpret $u(\vec{s})$ as a disutility function (or penalty function) for the objective. Without $u(s)$, we are just maximizing the worst-case profit without considering how likely that situation might be. The penalty $u(s)$ creates a distribution among the possible states, where the market-maker can then adjust by the multiplier w depending on his attitude towards risk. Therefore, $u(s)$ represents a penalty function with some informative distribution that allows the market-maker to potentially be less risk-averse, since we are not just analyzing the worst-case profit scenario.

Q2 (Theoretical): The disadvantage of the call auction model is that it cannot inform bidders whether or not their bids are accepted until the market closes. This is undesirable since sometimes bidders want to know the results of their bids immediately so that they can modify bids and submit again. In practice, the market is usually implemented in an online version defined as follows. Consider solving the optimization problem whenever the kth bidder submits a bid (for scalar x_k and vector $\vec{s}$):

$$\begin{array}{lll} \max_{x_k, s} & \pi_k x_k + u(s) & \\ \text{subject to} & a_k x_k + s_i = b_i - q_i^{k-1} & \forall\, i \in \{1, \dots m\} \\ & 0 \leq x_k \leq 1 & \\ & s_i \geq 0 & \forall\, i \in \{1, \dots, m\} \end{array}$$

where $q_i^{k-1} = \sum_{j=1}^{k-1} a_{ij}\overline{x}_j$ represents the amount of i that is already allocated before the kth bidder arrives. Note that only scalar x_k and vector $\vec{s}$ are variables. Assume again that $u(\cdot)$ is increasing and strictly concave. Write down the first-order KKT conditions, and argue why this online problem may be solved efficiently when compared to the offline problem.

Solution: Solving for $\vec{s}$ yields $s_i = b_i - q_i^{k-1} - a_{ik}x_k$ for all i. Substituting into the problem, we have

$$\begin{array}{lll} \max_{x_k,s} & \pi_k x_k + u(b_i - q_i^{k-1} - a_{ik}x_k) & \forall\ i \in \{1,\dots m\} \\ \text{subject to} & 0 \leq x_k \leq 1 & \\ & b_i - q_i^{k-1} - a_{ik}x_k \geq 0 & \forall\ i \in \{1,\dots,m\}. \end{array}$$

The Lagrangian for every i value is

$$L(x_k, \lambda_i, \mu_i) = \pi_k x_k + u(b_i - q_i^{k-1} - a_{ik}x_k) - \lambda_1(x_k - 1) + \lambda_2 x_k - \sum_{i=1}^{m} \mu_i(b_i - q_i^{k-1} - a_{ik}x_k)$$

where $\lambda_1, \lambda_2, \mu_i \geq 0$ are the Lagrange multipliers associated with the constraints $x_k \leq 1, x_k \geq 0$ and $\vec{s} \geq 0$. Compute the gradient

$$\nabla_{x_k} L(x_k, s) = \pi_k - \sum_{i=1}^{m} a_{ik}u'(b_i - q_i^{k-1} - a_{ik}x_k) - \lambda_1 + \lambda_2 - \sum_{i=1}^{m} \mu_i a_{ik} \qquad (\lambda_1, \lambda_2 \geq 0).$$

The KKT conditions (adding the complementary conditions) are then

$$\boxed{\begin{array}{rl} \pi_k - \sum_{i=1}^{m} a_{ik}u'(b_i - q_i^{k-1} - a_{ik}x_k) - \lambda_1 + \lambda_2 - \sum_{i=1}^{m} \mu_i a_{ik} x_k = 0 & \text{for } i \in \{1,\dots,m\} \\ \sum_i u'(b_i - q_i^{k-1} - a_{ik}x_k) = 1 & \text{prices sum to 1} \\ \lambda_1(x_k - 1) = 0 & \lambda_1, \lambda_2, \mu_i \geq 0 \\ \lambda_2 x_k = 0 & \\ \sum_{i=1}^{m} \mu_i a_{ik} = 0. & \end{array}}$$

The problem can be solved efficiently when compared to the offline problem, because the method uses a near-closed form solution (as shown in Lecture Note #18) and therefore is faster; in particular, it is not an iterative algorithm. The main idea of the online approach is to ensure that we can accept/reject a bid as soon as we receive it without having to wait for *all* bids (which is computationally expensive).

Q4 (Numerical): Now suppose there is a good estimate of the total n bidders in the market. Then we wait for the first k bidders to arrive and solve the linear program

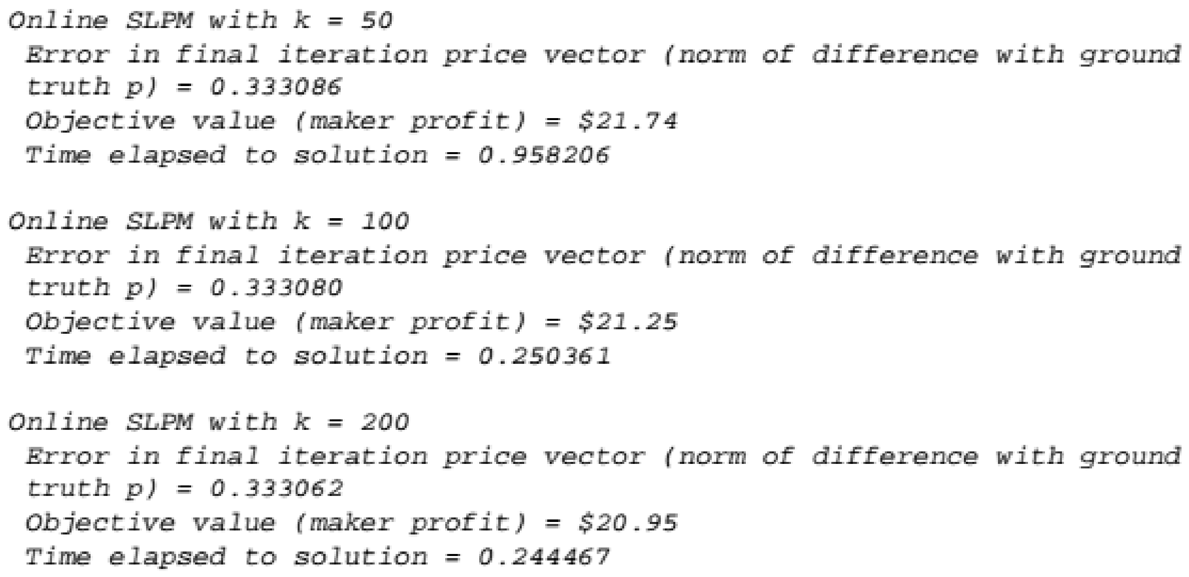
$$\begin{array}{lll} \max_{x_1,\dots,x_k} & \sum_{j=1}^{k} \pi_i x_i & \\ \text{subject to} & \sum_{j=1}^{k} a_{ij}x_j \leq \frac{k}{n} b_i & \forall\ i \in \{1,\dots m\} \\ & 0 \leq x_k j \leq 1 & \forall\ j \in \{1,\dots k\}. \end{array}$$

Then using the *dual prices* (call them $\overline{y}^k$) of the partial LP, we have

$$x_j = \begin{cases} 1 & \text{if } \pi_j > a_j^\intercal \overline{y}^k \\ 0 & \text{else.} \end{cases}$$

Let $n = 10,000$. We run the online SLPM algorithm using the same simulated bidding data for $k = 50, 100, 200$ to judge sensitivity with respect to iteration.

Solution: For the resource allocation on 10,000 bids, let $\vec{b} = (1000, \ldots, 1000).^\intercal$ The true price vector $\vec{p}_{\text{true}}$ was initialized as a random vector with entries $0 \leq p_i \leq 1$ such that $\sum_{i=1}^m p_i = 1$. We wait for the first k bids, then solve the SLPM problem to obtain the optimal dual price p. The subsequent $j > k$ bids are then accepted if their values are sufficiently high ($\pi_j > a_j^\intercal p$) and resources are still available ($a_j \leq b - \sum_{i=1}^{j-1} a_i x_i$). If $k = n$, the pure Offline SLPM model is solved. We then have the following output ($w = 0.812$) for the Online SLPM process:



```
Online SLPM with k = 50
 Error in final iteration price vector (norm of difference with ground
 truth p) = 0.333086
 Objective value (maker profit) = $21.74
 Time elapsed to solution = 0.958206

Online SLPM with k = 100
 Error in final iteration price vector (norm of difference with ground
 truth p) = 0.333080
 Objective value (maker profit) = $21.25
 Time elapsed to solution = 0.250361

Online SLPM with k = 200
 Error in final iteration price vector (norm of difference with ground
 truth p) = 0.333062
 Objective value (maker profit) = $20.95
 Time elapsed to solution = 0.244467
```

As demonstrated, the error $\|p_{\text{true}} - \vec{p}\|$ between the outputted price vector and $\vec{p}$ decreases (though only *very* marginally) as k varies from 50 to 200, and the market maker's profit decreases incrementally as well. With each iteration, more information is incorporated into determining p, and consequently error should decrease; however we see that within this range of k values the differences in the computed price vector and the objective function value are not dramatic. The standard Offline SLPM model produced a profit of \$20.64 given the parameters above. Therefore the online model essentially matches the offline standard by iteration $k = 200$, which is quite efficient performance.