Online Appendix to: Optimizing Prices in Descending Clock Auctions†

TRI-DUNG NGUYEN, University of Southampton, Schools of Mathematics and Management
TUOMAS SANDHOLM, Carnegie Mellon University, Computer Science Department *Corresponding author

APPENDIX

A.1. A Dynamic Programming Model for Optimal Price Setting in DCA

In each round of the descending clock auction, the auctioneer needs to offer each active bidder a price, i.e., to do Step 2.1. of Algorithm 1. Here we show a dynamic programming model that the optimal set of offer prices should solve.

Let $V(m, \mathcal{S}, \boldsymbol{u}, \boldsymbol{l})$ be the minimum expected payment that the auctioneer needs to pay to the bidders in a descending clock auction with m rounds, with a set of active bidders $\mathcal{S}$, with upper bounds $\boldsymbol{u}$ and lower bounds $\boldsymbol{l}$ within which the bidders' valuations lie. Let ξ be a realization of the bidders' values. For any offer prices $\boldsymbol{p}$ in the first round, the state of the auction by the end of that first round will be as follows.

— The number of rounds left will be $(m-1)$.
— The remaining active bidders will be $\mathcal{S}(\boldsymbol{p}, \xi)$. This includes bidder i if the offer price p_i is no smaller than the bidder's value ξ_i, i.e., $p_i \geq \xi_i$.
— A new vector of upper bounds $\boldsymbol{u}(\boldsymbol{p}, \xi)$ which updates the upper bound of any remaining active bidder i to x_i.
— Unchanged lower bounds $\boldsymbol{l}$.

The minimum expected value that the auctioneer needs to pay under the new state of the auction will be $V(m-1, \mathcal{S}(\boldsymbol{p}, \xi), \boldsymbol{u}(\boldsymbol{p}, \xi), \boldsymbol{l})$. Thus, the auctioneer's problem in the first round is to choose $\boldsymbol{p}$ that minimizes the expectation of $V(m-1, \mathcal{S}(\boldsymbol{p}, \xi), \boldsymbol{u}(\boldsymbol{p}, \xi), \boldsymbol{l})$. We have the Bellman optimality equation

$$V(m, \mathcal{S}, \boldsymbol{u}, \boldsymbol{l}) = \min_{\boldsymbol{p}} \quad E[V(m-1, \mathcal{S}(\boldsymbol{p}, \xi), \boldsymbol{u}(\boldsymbol{p}, \xi), \boldsymbol{l})].$$

Solving this dynamic program would be extremely difficult. In fact, just finding $V(m, \mathcal{S}, \boldsymbol{u}, \boldsymbol{l})$ for the case $m = 1$ would be very difficult as shown in a simple case below.

A.2. Optimal Price Setting in the Last Round with Recourse Action

Consider the problem of setting prices in the final round of a descending clock auction. Assume that the actual bid values are uniformly distributed random variables, i.e., $\xi_i \sim U[l_i, u_i]$, where $(l_i, u_i), i = 1, \ldots, n$ are known. Let $\boldsymbol{p}$ be a vector of prices that the auctioneer offers to the bidders. Given the offer prices, the bidders might accept or reject the offers. The auctioneer then updates the best upper bounds on the bids values, that is, upper bounds for accepting bidders will be updated to the offer prices while those of rejected bidders will remain unchanged. The auctioneer chooses T bids with the smallest updated upper bounds and pays each of these bidders those prices. Since the bidders' values are random variables, the acceptance of the bidders for each set of offer prices $\boldsymbol{p}$ will also be stochastic, so the final payment is stochastic. We consider the problem of finding the optimal offer prices $\boldsymbol{p}$ such that the expected final payment is minimized. Here expectation is taken over the randomness of the bidders' valuations.

DOI 10.1145/2600057.2602869 http://doi.acm.org/10.1145/2600057.2602869

For convenience in notation, we perform a linear transformation on the price vectors $\boldsymbol{p}$ to $\boldsymbol{x}$ where $x_i = \frac{p_i - l_i}{u_i - l_i}$, that is, $x_i \in [0,1]$ can be interpreted as the target chance of acceptance for bidder i. We also have $p_i = l_i + x_i(u_i - l_i)$. Let us denote by $f(\boldsymbol{x})$ the stochastic payment.

Let us first consider the simple case where $T = 1$ and $n = 2$. Here the payment is $min(u_1, u_2)$ if both bidders reject the offers, $min(p_1, p_2)$ if both of them accept, and $p_i, i = \{1, 2\}$ if only bidder i accepts the offer. The probability for each of these four events can be calculated as functions of $\boldsymbol{x}$. For example, the chance of rejecting both offers is $(1 - x_1)(1 - x_2)$. Putting all of these together, we have

$$f(\boldsymbol{x}) = \begin{cases} min(u_1, u_2), & \text{w.p. } (1-x_1)(1-x_2), \\ min(l_1 + x_1(u_1 - l_1), l_2 + x_2(u_2 - l_2)), & \text{w.p. } x_1 x_2, \\ l_1 + x_1(u_1 - l_1), & \text{w.p. } x_1(1-x_2), \\ l_2 + x_2(u_2 - l_2), & \text{w.p. } (1-x_1)x_2. \end{cases}$$

The expected payment is

$$\begin{aligned} E[f(\boldsymbol{x})] \;=\; & (1-x_1)(1-x_2)min(u_1,u_2) + x_1x_2min(l_1 + x_1(u_1-l_1), l_2 + x_2(u_2-l_2)) + \\ & x_1(1-x_2)(l_1 + x_1(u_1-l_1)) + (1-x_1)x_2(l_2 + x_2(u_2-l_2)). \end{aligned}$$

The problem of determining the optimal offer prices can therefore be formulated as

$$\begin{aligned} \min_{x_1,x_2} \quad & E[f(\boldsymbol{x})] \\ \text{s.t.} \quad & 0 \le x_i \le 1, \forall i = 1, 2, \end{aligned}$$

which is a non-convex quadratic optimization problem. If we extend the problem to the case $n > 2$, the problem becomes a polynomial optimization problem as follows;

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \sum_{\mathcal{S} \subset \mathcal{N}, \mathcal{S} \neq \emptyset} \left[\prod_{i \in \mathcal{S}} x_i \prod_{i \notin \mathcal{S}} (1 - x_i) \min_{i \in \mathcal{S}} \{l_i + x_i(u_i - l_i)\} \right] + \prod_{i \in \mathcal{N}} (1 - x_i) \min_{i \in \mathcal{N}} u_i \\ \text{s.t.} \quad & 0 \le x_i \le 1, \forall i = 1, \ldots, n, \end{aligned}$$

which is very difficult to solve. Notice that we have considered only the simple case of $T = 1$ and also considered finding the optimal decision in the last round only.

A.3. Proof of Proposition 2.5

PROOF. For each $m \in \{0, 1, \ldots, n\}$ and for each budget $B \ge 0$, let us define

$$\begin{aligned} V(m, B) = \min_{\boldsymbol{p}} \quad & \sum_{i=1}^{m} F_i(p_i) p_i, \\ \text{s.t.} \quad & \sum_{i=1}^{m} \omega_i F_i(p_i) \ge B, \\ & p_i \in \{P_{i1}, \ldots, P_{i,k}\}, \forall i = 1, \ldots, m. \end{aligned}$$

Then we have

$$\begin{aligned} V(m, B) = \min_{p_m} \quad & F_m(p_m) p_m + V(m-1, B - F_m(p_m)), \\ \text{s.t.} \quad & p_m \in \{P_{m1}, \ldots, P_{m,k}\}, \end{aligned} \tag{16}$$

where $V(0, B) = 0, \forall B$. Suppose $F_j(p_j)$ receives one of $(L + 1)$ values in the set $\{0, 1/L, \ldots, 1\}$. Then we can calculate $V(1, B)$ for all $B \in \{0, 1/L, \ldots, 1\}$. If we knew $V(m-1, B), \forall B \in \{0, 1/L, \ldots, m-1\}$, then we can plug this in into Formulation 16

and obtain $V(m, B)$ by taking K calculations for $F_m(p_m)p_m + V(m-1, B - F_m(p_m))$ for each $p_m \in \{P_{m1}, \ldots, P_{m,k}\}$ and then choose the minimum, i.e., $2K$ operations in total. To obtain $V(m, B)$ for all possible $B \in \{0, 1/L, \ldots, m\}$, we would need to repeat this Lm times, which means the total operations incurred for each m is $2KLm$. Summing this for all $m \in \{1, \ldots, n\}$ would require $KLn(n-1)$ operations. Thus the complexity of the algorithm is O(KLn^2). □

A.4. Descending Clock Auctions using Optimized Price Setting for the Homogeneous Setting

ALGORITHM 2: A DCA Framework using Optimal Price Setting for the Homogeneous Setting

Input: A set of sellers $\mathcal{N} = \{1, \ldots, n\}$ with goods $\{G_1, \ldots, G_n\}$, an auctioneer with a target T. A target number of rounds allowed m. Initial valuation estimates v_i.

Output: A set of feasible sellers $\mathcal{A} \subset \mathcal{N}$, i.e., $|\mathcal{A}| = T$, and the corresponding offer price vector $\boldsymbol{p}$ that aims to minimize the expected payment.

1. Let the set of active bidders be $\mathcal{A}^{(r)} = \mathcal{N}$;

for *round* $r = 1...m$ **do**

2.1. Set the target number of accepting bidders $T^{(r)} = n^{(r)} - (n^{(r)} - T)/(m - r)$ where $n^{(r)} = |\mathcal{A}^{(r)}|$ and solve Model 9 to find a vector of prices $\boldsymbol{p}$ to offer the bidders;

2.2. Find the set of rejected offers $\mathcal{R}$;

if $|(\mathcal{A}^{(r)} \backslash \mathcal{R}| \geq T$ **then**

2.2.1. $\mathcal{A}^{(r+1)} \leftarrow \mathcal{A}^{(r)} \backslash \mathcal{R}$;

2.2.2. Update the distributions of the bidders' valuations using Formulation 11;

else

2.2.3. Enter the adjustment round in Step 3;

end

end

3. Adjust the prices for bidders in the last round to meet the target by solving Formulation 12;

4. Pay winning bidders the offer prices;

A.5. Descending Clock Auctions using Optimized Price Setting for Incentive Auctions

ALGORITHM 3: A DCA Framework using Optimal Price Setting for Incentive Auctions

Input: A set of stations $\mathcal{N} = \{1, \ldots, n\}$, an auctioneer with a feasibility function $F : 2^{\mathcal{N}} \rightarrow \{0, 1\}$. A target number of rounds allowed m. Initial valuation function estimates v_i.

Output: A set of feasible stations to reject $\mathcal{R} \subset \mathcal{N}$, i.e. $F(\mathcal{R}) = 1$, and the corresponding offer price vector $\boldsymbol{p}$ that aims to minimize the expected payment on the remaining stations.

1. Set the initial prices $\boldsymbol{p}$ at the reserves. Let the set of rejected bidders be $\mathcal{R}^{(r)} = \emptyset$;

for *round* $r = 1...m$ **do**

2.1. Set the target number of accepting bidders $T^{(r)} = n^{(r)} - (n^{(r)} - T)/(m - r)$ where $n^{(r)} = |\mathcal{A}^{(r)}|$ and solve Model 9 to find a vector of prices $\boldsymbol{p}$ to offer the bidders;

2.2. Find the set of rejected offers $\mathcal{R}$;

if $F(\mathcal{R}^{(r)} \cup \mathcal{R}) = 1$, i.e. via solving 17, **then**

2.2.1. $\mathcal{R}^{(r+1)} \leftarrow \mathcal{R}^{(r)} \cup \mathcal{R}$;

2.2.2. Update the distributions of the bidders' valuations using Formulation 11;

else

2.2.3. Enter the final Step 3;

end

end

3. Set all remaining bidders $\mathcal{N} \backslash \mathcal{R}^{(r)}$ as winners and pay them their offer prices;

A.6. Approximation Method for the Multi-Round Case

First, assuming that the auctioneer has only one round left. Then the optimal prices to offer to the bidders will be the solution of Model 9 for the continuous case (or 10 for the discrete case). Now, given that the auctioneer has multiple rounds to do price discovery, he would not offer these 'optimal prices' right away. Instead, a set of higher prices will be offered first to learn more about the bidders' valuations and to update the bounds.

A simple way that the auctioneer can do this is to discretize the prices into m equal intervals between the uppers bounds and $\boldsymbol{p}^*$ and offer these to the bidders sequentially until feasibility does not hold.

A better way is to do this dynamically as shown in Algorithm 4. Here, after solving Model 9 (or 10) in Step 1, the auctioneer can offer a guess $p_i = \frac{u_i+(m-1)p_i^*}{m}$ to bidder i and see how the bidder responds. This price is obtained under the expectation that the offer price in the next m rounds will be distributed evenly within the range $[p_i^*, u_i]$. Notice, however, that once the auctioneer has offered the prices to the bidders and received their responses to form the new state of the auction, the auctioneer now has better information and can repeat Step 2.1 of Algorithm 1 to find the new set of offer prices, that is, to run Algorithm 4 again with the updated information. Formally:

ALGORITHM 4: Finding Offer Prices in Round m

Input: Current round r, a current set of active bidders $\mathcal{A}^{(r)}$, most up-to-date valuation estimates v_i.
Output: An offer price vector $\boldsymbol{p}$.
1. Solve Model 9 to obtain the optimal offer prices $\boldsymbol{p}^*$ as if this were the last round;
2. Divide the range $[p_i^*, u_i]$ into m equal intervals and set the actual offer prices $p_i = \frac{u_i+(m-1)p_i^*}{m}$;

A.7. Interference Constraints in Repacking and Feasibility Checking

There are two csv data files on engineering constraints available on the FCC web site [FCC 2013]:

- A domain file called "Domain-2013July15.csv", of size 306KB, that specifies the feasible channels for each station.
- An interference file called "Interference-Paired-2013July15.csv", of size 6219KB, that specifies the interference constraints that the repacking must meet. This includes:
  - Pairs of (station, station) that must not be assigned to the same channel (among a given list of channels).
  - Pairs of (station, station) that must not be assigned to adjacent channels (among a given list of channels).

The average number of feasible channels that each station can be allocated to is 44.15 (out of 49 channels) with most of the channels being freely allocated to any available channels. However, some stations only have a few feasible channels (that is, there are stations with only three possible channel assignments). There are 2.9×10^6 constraints requiring pairs of stations that are not to be allocated in the same or adjacent channels. Although this is smaller than $2mn^2 = 493 \times 10^6$ in the worst case, i.e., when interference matrices are fully dense, it is still a large number.

Let $\mathcal{S}$ be a set of stations that needs to be repacked into a list of channels in set $\mathcal{C}$. We use i, j as indices for stations and use k as indices for channels. Let $\mathcal{C}_i \subset \mathcal{C},\ i \in \mathcal{S}$, be the list of feasible channels to station i. Let $\mathcal{I}_c$ be the list of triplets (i, j, k) such that

stations i and j cannot be assigned to the same channel k. Let $\mathcal{I}_a$ be the list of triplets (i, j, k) such that stations (i, j) cannot be assigned to channel $(k, k+1)$ respectively. Data for $\mathcal{C}_i$, $\mathcal{I}_c$ and $\mathcal{I}_a$ are available from the domain file and the interference-paired file on the FCC web site [FCC 2013].

From a given list of channels $\mathcal{C}$, we say the set $\mathcal{S}$ of stations is feasible with respect to $\mathcal{C}$ if the stations can be packed into the channels without violating any of the constraints. Let $\mathcal{P}(\mathcal{C})$ denote the set of all subsets of stations that can be feasibly packaged into channels in $\mathcal{C}$.

Let z_{ik} be a binary variable that indicates whether station i is assigned to channel k. We say $\boldsymbol{z}$ is an assignment to the repacking problem. For $\boldsymbol{z}$ to be feasible, we need the following: (a) all the indicator variables z_{ik} are binary, (b) each station is assigned to exactly one channel, and (c) no pairs of stations that might interfere with each other can be assigned to the same or adjacent channels. The set of feasible assignments $\mathcal{P}(\mathcal{C})$ is therefore defined as

$$\mathcal{P}(\mathcal{C}) \;=\; \left\{ \boldsymbol{z}: \quad \begin{array}{l} z_{ik} \in \{0,1\}, \forall i \in \mathcal{S} \text{ and } k \in C_i,\ \sum_{k \in C_i} z_{ik} = 1, \forall i \in \mathcal{S}, \\ z_{ik} + z_{jk} \leq 1, \forall (i,j,k) \in \mathcal{I}_c,\ z_{ik} + z_{jk+1} \leq 1, \forall (i,j,k) \in \mathcal{I}_a \end{array} \right\}. \quad (17)$$

There are a large number—up to 2.9×10^6—of constraints requiring pairs of stations not to be allocated in the same or adjacent channels. This makes checking the assignment feasibility very challenging for the full problem when all 2177 stations are rejected. In our experiments, however, the largest number of stations being rejected among all the instances tested is less than 1000 and hence CPLEX can still handle the feasibility problem. The feasibility problem does not involve an objective function and hence is much easier to solve than the winner determination problem in a VCG setting.

In the experiments, we used CPLEX to solve the repacking feasibility problem. We could also use a satisfiability (SAT) formulation for this purpose as has been done by Leyton-Brown [2013]. Our choice of CPLEX here was for the convenience of implementation and due to some special network structural properties of the repacking problem that CPLEX could exploit. However, a discussion on the comparison between the performance of SAT and CPLEX is out of the scope of this manuscript since our focus is on the price setting and not on computational method for solving the feasibility problem.