Towards Simple Polynomial-Time Optimal Auctions

Takehiro Oyakawa\(^1\) and Amy Greenwald\(^1\)

\(^1\)Department of Computer Science, Brown University, Providence, RI

\{oyakawa, amy\}@cs.brown.edu

November 16, 2016

Abstract

Border’s theorem gives us a way of solving mechanism design problems in polynomial time. This approach requires the use of linear programming machinery, the mechanics of which are significantly more complicated than the pointwise maximization approach that Myerson’s analysis of single-parameter mechanism design problems yields. However, Myerson’s approach, applied naively, yields an exponential-time algorithm. Motivated by the simplicity of Myerson’s analysis, we give a simple method for solving the usual one-good, many-bidders optimal auction problem in polynomial time. This method can be applied to both surplus and revenue maximization, yields symmetric outcomes in expectation, and does not rely on linear programming.

1 Introduction

In the classic single-good, sealed bid auction setting, Vickrey [10] showed that bidders are incentivized to bid their true valuation for the good being sold by requiring winners to pay the second highest bid. Consequently, Vickrey auctions maximize welfare. Building on this idea, Myerson [8] showed that the auctioneer can maximize total expected revenue by adding to this design a per-bidder reserve price. In such auctions, one may naturally ask: what is the expected revenue of the auction, what is the probability of allocating a good to a bidder (modulo ties), and how much should each bidder expect to pay?

To answer these questions, one could apply Myerson’s analysis in a straightforward manner, but this would require an exponential number of variables and constraints. Using only a polynomial number of feasibility constraints and an ellipsoid-style method, one could compute the quantities of interest in polynomial time. (See, for example, Cai et al. [3].) This approach, however, relies on linear programming machinery, and is neither as simple nor as intuitive as Myerson’s approach.

In this work, we put forth an alternative solution, which is both fast and intuitive. First off, our method estimates total expected revenue in polynomial time. That, however, can be done via Monte Carlo methods. More importantly, our method also yields allocation probabilities, rather than a fixed allocation rule given by a serially dictatorial scheme, which prescribes preferences over which bidders are allocated in the event ties occur. This guarantees that allocation probabilities, and hence expected payments, for any two symmetric bidders, are equivalent. For example, suppose we had two bidders, Alice and Bob, and they each placed a bid of $10 on a good being sold. Since their bids are equivalent, the ex-ante probability that either Alice or Bob wins is half. Ultimately, the auctioneer must implement a tie-breaking rule (i.e., a serially dictatorial allocation scheme) in order to determine who wins the good; say, it implements lexicographic tie breaking. After ties are
broken (i.e., \textit{ex-post}), the probability of winning is no longer equivalent for the bidders: Alice wins with probability 1, and Bob with probability 0. In this work, are interested in ex-ante allocation probabilities: i.e., the probability of winning before any tie breaking.

\textbf{Related Work} Vickrey [10] showed that auctions in which the highest bidder wins and pays the second-highest bid incentivize bidders to bid truthfully. Myerson [8] showed that in the single-parameter setting, with the usual quasi-linear utility function involving linear payments, total expected revenue is maximized by a Vickrey auction with reserve prices based on distributions of bidder valuations. Understanding how much data are needed to understand valuation distributions well enough to guarantee close to optimal expected revenue has been studied by Cole and Roughgarden [4]. In principle, revenue maximization for some mechanism design problems, such as the garden [4]. In principle, revenue maximization for some mechanism design problems, such as the second-highest bid incentivize bidders to bid truthfully. Myerson [8] showed that in the single-parameter setting, with the usual quasi-linear utility function involving linear payments, total expected revenue is maximized by a Vickrey auction with reserve prices based on distributions of bidder valuations. Understanding how much data are needed to understand valuation distributions well enough to guarantee close to optimal expected revenue has been studied by Cole and Roughgarden [4]. In principle, revenue maximization for some mechanism design problems, such as the garden [4]. In principle, revenue maximization for some mechanism design problems, such as the second-highest bid incentivize bidders to bid truthfully. Myerson [8] showed that in the single-parameter setting, with the usual quasi-linear utility function involving linear payments, total expected revenue is maximized by a Vickrey auction with reserve prices based on distributions of bidder valuations. Understanding how much data are needed to understand valuation distributions well enough to guarantee close to optimal expected revenue has been studied by Cole and Roughgarden [4]. In principle, revenue maximization for some mechanism design problems, such as the garden [4].

\section{2 Model and Background}

An auctioneer would like to sell a good to one of \( n \) bidders. Each bidder \( i \in N = \{1, \ldots, n\} \) has a private valuation \( 0 \leq v_i \in T_i \) that is independently drawn from some distribution \( F_i \). Let \( T = T_1 \times \cdots \times T_n \) be the set of all possible valuation vectors (profiles), and let \( F = F_1 \times \cdots \times F_n \) be the distribution over valuation vectors \( v = (v_1, \ldots, v_n) \in T \). Let \( b = (b_1, \ldots, b_n) \in \mathbb{R}^n \) be a vector of bids, where the \( i \)-th entry \( b_i \) is bidder \( i \)'s bid. For \( y \in \{v, b\} \), we use the notation \( y = (y_i, y_{-i}) \), where \( y_{-i} = (y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_n) \). Similarly, we let \( T_{-i} = \prod_{j \neq i} T_j \) and \( F_{-i} = \prod_{j \neq i} F_j \).

Given a vector of reports \( b \), a mechanism determines an allocation rule \( x(b) \in [0, 1]^n \), where bidder \( i \)'s allocation probability is \( x_i(b_i, b_{-i}) \), together with a payment rule \( p(b) \in \mathbb{R}^n \geq 0 \), where bidder \( i \)'s payment is \( p_i(b_i, b_{-i}) \). We define bidder \( i \)'s quasi-linear utility function as

\[ u_i(b_i, b_{-i}) = v_ix_i(b_i, b_{-i}) - p_i(b_i, b_{-i}). \quad (1) \]

Next, we formalize the usual constraints imposed on optimal auction design. Because we restrict our attention to incentive compatible auctions, where it is optimal to bid truthfully, hereafter, we write \( x_i(v_i, v_{-i}) \) instead of \( x_i(b_i, b_{-i}) \), and \( p_i(v_i, v_{-i}) \) instead of \( p_i(b_i, b_{-i}) \).

We introduce \textit{interim allocation} and \textit{interim payment} variables, respectively: \( \hat{x}_i(v_i, \cdot) = \mathbb{E}_{v_{-i}}[x_i(v_i, v_{-i})] \) and \( \hat{p}_i(v_i, \cdot) = \mathbb{E}_{v_{-i}}[p_i(v_i, v_{-i})] \). These variables comprise the interim allocation rule \( \hat{x}(v) \in [0, 1]^n \) and \( \hat{p}(v) \in \mathbb{R}^n \geq 0 \).

We call a mechanism \textbf{(Bayesian) incentive compatible (IC)} if utility is maximized by truthful reports in expectation: \( \forall i \in N \) and \( \forall v_i, w_i \in T_i \), \( v_i\hat{x}_i(v_i) - \hat{p}_i(v_i) \geq v_i\hat{x}_i(w_i) - \hat{p}_i(w_i) \). A mechanism is \textbf{Individually rational (IR)} if it insists on non-negative utilities in expectation: \( \forall i \in N \) and \( \forall v_i \in T_i \), \( v_i\hat{x}_i(v_i) - \hat{p}_i(v_i) \geq 0 \). We say a mechanism is \textbf{ex-post feasible (XP)} if it never overallocates: \( \forall v \in T, \sum_{i=1}^n x_i(v_i, v_{-i}) \leq 1 \).

Finally, we require that \( 0 \leq x_i(v_i, v_{-i}), \hat{x}_i(v_i) \leq 1, \forall i \in N, \forall v_i \in T_i \) and \( \forall v_{-i} \in T_{-i} \).
2.1 Myerson’s Payment Analysis, and Pointwise Maximization

Myerson’s payment theorem, which takes as a starting point the bidders’ utility functions (i.e., Equation (1)), was expressed assuming that bidders drew valuations from continuous distributions. Here, we apply his analysis to discrete valuations.

Specifically, we assume the distribution of values is drawn from the discrete type space \( T_i = \{z_{i,k} : 1 \leq k \leq M_i \} \), of cardinality \(|T_i| = M_i\), where \( z_{i,j} < z_{i,k} \) for \( j < k \), and we let \( z_{i,|T_i|+1} = z_{i,|T_i|} \). We also assume the probability of type \( z_{i,k} \in T_i \) is given by cumulative distribution function \( F_i(z_{i,k}) \) and corresponding probability mass function \( f_i(z_{i,k}) \). In addition, \( f_{-i}(v_{-i}) \) is the probability mass function of \( v_{-i} \in T_{-i} \).

**Theorem 2.1** (\cite{8}). Assume bidders’ utilities take the form of Equation (1). An optimal mechanism is IC and IR iff for all \( i \in N \): the allocation rule \( \hat{x}_i \) is monotone, i.e., \( \forall v_i \geq w_i \in T_i, \hat{x}_i(v_i) \geq \hat{x}_i(w_i) \); and the payment rule \( \hat{p}_i \) is given by: \( \forall z_{i,\ell} \in T_i, \)

\[
\hat{p}_i(z_{i,\ell}) = z_{i,\ell} \hat{x}_i(z_{i,\ell}) - \sum_{j=1}^{\ell-1} (z_{i,j+1} - z_{i,j}) \hat{x}_i(z_{i,j}).
\]  

Note that \( \hat{p}_i(z_{i,1}) = z_{i,1} \hat{x}_i(z_{i,1}) \).

Using Myerson’s analysis, we can see that if maximizing total expected surplus, \( \mathbb{E}_{v \sim F}[v \cdot x(v)] \), were the objective, then we can do so pointwise, solving for an optimal allocation for each \( v \) in turn, subject only to ex-post feasibility. This allocation rule can be supported with Myerson’s payment rule, thereby ensuring IC and IR. Algorithm 1 describes this pointwise approach in detail, where, when maximizing total expected surplus is the objective, each \( L_i \) is the identity function. Notice that Algorithm 1 preserves tie-breaking probabilities in allocation terms, and does not employ an allocation that is serially dictatorial, as described by Svensson \cite{9}, thus preserving symmetry in the final outcome.

Algorithm 1 is exponential in runtime. Let the cardinality of the largest type space be \( \mathcal{R} = \max_{i \in N}|T_i| \). There are \( O(\mathcal{R}^n) \) valuation vectors in \( T \). For each valuation vector, determining allocations is \( O(n) \), so determining all allocations is \( O(n\mathcal{R}^n) \). Each \( \hat{x}_i(z_{i,\ell}) \) is computed in \( O(\mathcal{R}^{n-1}) \), so determining all interim allocations takes \( O(n\mathcal{R}^n) \). Each \( \hat{p}_i(z_{i,\ell}) \) is computed in \( O(\mathcal{R}) \), so determining all interim payments takes \( O(n\mathcal{R}^2) \). Computing \( S \) is done in \( O(n\mathcal{R}) \). Computing \( R \) is done in \( O(n\mathcal{R}) \). Therefore, the complexity of Algorithm 1 is \( O(n\mathcal{R}^2) \).

2.2 Applying Pointwise Maximization to Revenue

In his seminal work on optimal auction design, Myerson proved that expected revenue can be expressed as something called expected virtual surplus, which he defined in terms of virtual valuations. Using our notation, we define virtual valuations for discrete distributions as follows:

\[
\psi_i(z_{i,k}, z_{i,k+1}) = z_{i,k} - (z_{i,k+1} - z_{i,k}) \left(1 - \frac{F_i(z_{i,k})}{f_i(z_{i,k})}\right).
\]

We use the shorthand \( \psi_i(v_i) = \psi_i(z_{i,k}, z_{i,k+1}) \), where \( v_i = z_{i,k} \) for some \( 1 \leq k \leq |T_i| \). We assume our problem is regular, as in \cite{8}, so that \( \psi_i(z_{i,k+1}) > \psi_i(z_{i,k}) \) whenever \( z_{i,k+1} > z_{i,k} \).

Using the discrete version of Myerson’s payment formula (Theorem 2.1), and following a similar analysis to that of Myerson \cite{8}, we arrive at the following theorem:
Algorithm 1 Pointwise Maximizer

1: for all \( v \in T \) do  \hspace{1cm} \triangleright \text{Find allocations}
2: \hspace{1cm} for all \( i \in N \) do
3: \hspace{2cm} \( x_i(v_i, v_{-i}) \leftarrow 0 \)
4: \hspace{1cm} end for
5: \hspace{1cm} if any of the \( L_i(v_i) \)'s are positive then
6: \hspace{2cm} \( w \leftarrow \arg \max \{ L_i(v_i) \} \)
7: \hspace{2cm} for all \( i^* \in w \) do
8: \hspace{3cm} \( x_i^*(v_{i^*}, v_{-i^*}) \leftarrow 1/|w| \)
9: \hspace{2cm} end for
10: \hspace{1cm} end if
11: \hspace{1cm} end for
12: \hspace{1cm} for all \( i \in N \) do
13: \hspace{2cm} for \( \ell = 1 \) to \( |T_i| \) do
14: \hspace{3cm} \( \hat{x}_i(z_{i,\ell}) \leftarrow E_{v_{-i}}[x_i(z_{i,\ell}, v_{-i})] \)  \hspace{1cm} \triangleright \text{Compute interim allocations}
15: \hspace{3cm} \( \hat{p}_i(z_{i,\ell}) \leftarrow z_{i,\ell} \hat{x}_i(z_{i,\ell}) - \sum_{j=1}^{\ell-1} (z_{i,j+1} - z_{i,j}) \hat{x}_i(z_{i,j}) \)  \hspace{1cm} \triangleright \text{Compute interim payments}
16: \hspace{2cm} end for
17: \hspace{1cm} end for
18: \( S \leftarrow \sum_{i=1}^{n} E_{z_{i,\ell}}[z_{i,k} \hat{x}_i(z_{i,\ell})] \)  \hspace{1cm} \triangleright \text{Total expected surplus}
19: \( R \leftarrow \sum_{i=1}^{n} E_{z_{i,\ell}}[\hat{p}_i(z_{i,\ell})] \)  \hspace{1cm} \triangleright \text{Total expected revenue}
20: return \( S, R, \hat{x}, \hat{p} \)

Theorem 2.2 (§). Assume bidders’ utilities take the form of Equation (1). If a mechanism is IC and IR, then for all \( i \in N \),

\[
E_{z_{i,k} \sim F_i} [\hat{p}_i(z_{i,k})] = E_{z_{i,k} \sim F_i} [\psi_i(z_{i,k}) \hat{x}_i(z_{i,k})].
\]  \hspace{1cm} (4)

Theorem 2.2 lets us express the total expected revenue in terms of virtual surplus,

\[
\sum_{i \in N} E_{z_{i,k} \sim F_i} [\hat{p}_i(z_{i,k})] = \sum_{i \in N} E_{z_{i,k} \sim F_i} [\psi_i(z_{i,k}) \hat{x}_i(z_{i,k})]
\]

\[
= \sum_{i \in N} E_{v \sim F_i} [\psi_i(v_i) x_i(v_i, v_{-i})],
\]  \hspace{1cm} (5)

so maximizing Equation (5) can be done pointwise, solving for an optimal allocation for each \( v \) in turn. This procedure is again given by Algorithm 1 where, when maximizing total expected revenue is the objective, \( L_i \) is the virtual valuation function of bidder \( i \).

In maximizing Equation (5), bidders with negative virtual valuations will never be allocated. Since bidders must place a bid that maps to a non-negative virtual valuation in order to be allocated, it means that each bidder \( i \) has a \textbf{reserve price}, \( \psi_i^{-1}(0) \), which is the smallest bid she may place in order to possibly be allocated, and the smallest possible payment she must make when allocated.

3 On Ties

In this section, we describe how the expected revenue from a bidder can be described by a partition of calculations. This partitioning does not change the fact that calculating the expected revenue
takes an exponential amount of time. However, in the very special case of no ties, partitioning in this way leads to a polynomial-time calculation of virtual surplus (revenue).

Bidder \(i\) can only be allocated when the following criteria are met:

- \(i\) has placed a bid that meets \(i\)'s reserve price: \(\psi_i(v_i) \geq 0\), and
- \(i\)'s virtual valuation is at least as large as everyone else's: \(\psi_i(v_i) \geq \max_{j \in N \setminus i} \psi_j(v_j)\).

Let \(w(v)\) be the set of bidders that can potentially be allocated:

\[
 w(v) = \left\{ i : \psi_i(v_i) \geq 0, \psi_i(v_i) \geq \max_{j \in N \setminus i} \psi_j(v_j), \forall i \in N \right\}. \tag{7}
\]

The probability that a bidder is allocated depends on whether there are any ties or not:

\[
 x_i(v_i, v_{-i}) = \begin{cases} |w(v)|^{-1} & \text{if } i \in w(v) \\ 0 & \text{otherwise}. \end{cases} \tag{8}
\]

We know that \(i\) will not be allocated when she bids less than her reserve price; only valuations at least as large as the reserve price may be allocated. Let \(\tau_{i}^{\geq 0}\) be the set of bidder \(i\)'s valuations that at least match her reserve price:

\[
\tau_{i}^{\geq 0} = \{ z_{i,k} : \psi_i(z_{i,k}) \geq 0, \forall z_{i,k} \in T_i \}. \tag{9}
\]

Similarly, we know that bidder \(i\) is guaranteed not to be allocated when there exists any other bidder with virtual valuation larger than \(i\)'s. Specifically, in order for \(i\) to win with valuation \(z_{i,k}\), the other bidders virtual valuations must be such at most \(\psi_i(z_{i,k})\). Let \(\tau_{-i}^{\leq \psi_i(z_{i,k})}\) be the set of profiles other bidders may have where \(i\)'s virtual valuation, \(\psi_i(z_{i,k})\), is at least as large as any other:

\[
\tau_{-i}^{\leq \psi_i(z_{i,k})} = \left\{ v_{-i} : \max_{j \in N \setminus i} \psi_j(v_j) \leq \psi_{i,k}(z_{i,k}), \forall v_{-i} \in T_{-i} \right\}. \tag{10}
\]

We can now express the interim allocation \(\hat{x}_i(z_{i,k})\) as follows:

\[
\hat{x}_i(z_{i,k}) = \sum_{v_{-i} \in \tau_{-i}^{\leq \psi_i(z_{i,k})}} x_i(v_i, v_{-i}) f_{-i}(v_{-i}). \tag{11}
\]

Furthermore, we can partition the calculation of total expected revenue from bidder \(i\) as follows:

\[
\sum_{k=1}^{\lvert T_i \rvert} \psi_i(z_{i,k}) f_i(z_{i,k}) \hat{x}_i(z_{i,k}) = \sum_{z_{i,k} \in \tau_{i}^{\geq 0}} \psi_i(z_{i,k}) f_i(z_{i,k}) \hat{x}_i(z_{i,k}). \tag{12}
\]

This total expected revenue formula involves only a subset of \(T\). However, as indicated by Equation (12), this subset is still exponential in size. Next, we observe that when there are no ties, computing total expected revenue can be done in polynomial time.

Assuming no ties, when \(i\) wins, \(x_i(v_i, v_{-i}) = 1\). Let \(\tau_{-i}^{< \psi_i(z_{i,k})}\) be the set of profiles other bidders may have where \(i\)'s virtual valuation, \(\psi_i(z_{i,k})\), is strictly larger than any other:

\[
\tau_{-i}^{< \psi_i(z_{i,k})} = \left\{ v_{-i} : \max_{j \in N \setminus i} \psi_j(v_j) < \psi_{i,k}(z_{i,k}), \forall v_{-i} \in T_{-i} \right\}. \tag{13}
\]

\(^1\)We assume the seller is always willing to sell, so long as these conditions are met. That is, the seller has zero valuation for the good being sold. Otherwise, a simple extension of the criteria can be included to incorporate the seller’s valuation.
In this strict case, Equation \((11)\) becomes
\[
\hat{x}_i(z_{i,k}) = \sum_{v_{-i} \in \tau_{-i}^{< \psi_i(z_{i,k})}} f_{-i}(v_{-i}).
\] (14)

Finally, since we are working with independent and identically distributed random variables, the interim allocation calculation is no longer exponential:
\[
\sum_{v_{-i} \in \tau_{-i}^{< \psi_i(z_{i,k})}} f_{-i}(v_{-i}) = \prod_{j \neq i} \Pr(\psi_i(z_{i,k}) > \psi_j(z_{j,\ell}))
= \prod_{j \neq i} \sum_{z_{j,\ell} : \psi_i(z_{i,k}) > \psi_j(z_{j,\ell})} f_j(z_{j,\ell}).
\] (15)

In other words, assuming no ties, interim allocations can be computed in polynomial time. Consequently, we can also compute total expected revenue in polynomial time. We describe this procedure in Algorithm 2, where \(L_i\) is \(i\)'s virtual valuation function.

**Algorithm 2** Optimized Pointwise Maximizer (No Ties)

1: for all \(i \in N\) do
2:   \(\text{for all } z_{i,k} \in T_i \setminus \tau_{i,L_i}^{\geq 0}\) do \(\triangleright \tau_{i,L_i}^{\geq 0} = \{z_{i,k} : L_i(z_{i,k}) \geq 0, \forall z_{i,k} \in T_i\}\)
3:     \(\hat{x}_i(z_{i,k}) \leftarrow 0\)
4:   end for
5: for all \(z_{i,k} \in \tau_{i,L_i}^{\geq 0}\) do
6:   \(\hat{x}_i(z_{i,k}) \leftarrow \prod_{j \neq i} \sum_{z_{j,\ell} : L_i(z_{i,k}) > L_j(z_{j,\ell})} f_j(z_{j,\ell})\)
7: end for
8: for \(\ell = 1\) to \(|T_i|\) do \(\triangleright \text{Calculate interim payments}\)
9:   \(\hat{p}_i(z_{i,\ell}) \leftarrow z_{i,\ell} \hat{x}_i(z_{i,\ell}) - \sum_{j=1}^{\ell-1} (z_{i,j}+1 - z_{i,j}) \hat{x}_i(z_{i,j})\)
10: end for
11: end
12: \(S \leftarrow \sum_{i=1}^{n} E_{z_{i,\ell}} [z_{i,k} \hat{x}_i(z_{i,\ell})]\) \(\triangleright \text{Total expected surplus}\)
13: \(R \leftarrow \sum_{i=1}^{n} E_{z_{i,\ell}} [\hat{p}_i(z_{i,\ell})]\) \(\triangleright \text{Total expected revenue}\)
14: return \(S, R, \tilde{x}, \tilde{p}\)

Algorithm 2 is polynomial in runtime. For each bidder, setting \(\hat{x}_i(z_{i,k})\) to 0 for all \(z_{i,k} \in T_i \setminus \tau_{i,L_i}^{\geq 0}\) is \(O(\tilde{R})\). Computing \(\hat{x}_i(z_{i,k})\) is done in \(O(n \tilde{R})\). Doing so for all \(z_{i,k} \in \tau_{i,L_i}^{\geq 0}\) is \(O(n^2 \tilde{R}^2)\). Computing every \(\hat{p}_i(z_{i,\ell})\) is \(O(\tilde{R}^2)\). Therefore, determining interim allocations takes \(O(n^2 \tilde{R}^2)\), and determining interim payments takes \(O(n^3 \tilde{R}^2)\). Computing \(S\) is done in \(O(n \tilde{R})\). Computing \(R\) is done in \(O(n \tilde{R})\). Therefore, the complexity of Algorithm 2 is \(O(n^2 \tilde{R}^2)\).

**Remark 3.1.** When there are no ties, Algorithm 1 and Algorithm 2 both give the same output.

**Remark 3.2.** When there are no ties, and \(L_i\) is the identity function, Algorithm 2 provides a way of maximizing total expected surplus in polynomial time.

# 4 Perturbations to Virtual Valuations

At this point, we have observed that the total expected revenue calculation is polynomial-time when there are no ties. It remains, then, to show how ties can also be handled in polynomial-time. This is the subject of the rest of the paper.
Similar to Hartline’s perturbation approach \[6\], our strategy for handling ties will be to transform any set of virtual valuations into one that is guaranteed to have no ties, without changing the allocation function too much. In this section, we show that we can modify virtual valuations slightly so that (i) any virtual valuation larger (smaller) than any other virtual valuation continues to remain larger (smaller), and (ii) any virtual valuation above (below) the reserve price remains above (below) the reserve price. This method will yield, in expectation, allocations described by Algorithm \[1\] which preserves tie-breaking probabilities and does not employ a serially dictatorial allocation scheme. However, if one is interested in an outcome that uses a serially dictatorial allocation method, then the method described in this section can be adapted to do so.

**Lemma 4.1.** There exists an \(\epsilon_A \in \mathbb{R}_{>0}\) such that for all \(r_{i,k} \in [0,1]\), if \(\psi_i(z_{i,k}) > \psi_j(z_{j,\ell})\), then \(\psi_i(z_{i,k}) + r_{i,k}\epsilon_A > \psi_j(z_{j,\ell}) + r_{j,\ell}\epsilon_A\), for all \(i, j \in N\), \(1 \leq k \leq |T_i|\), \(1 \leq \ell \leq |T_j|\).

**Proof.** Consider any set of virtual valuations \(\psi_1 > \psi_2 > \psi_3\). For \(c, d \in \{1, 2, 3\}\), let \(\delta_{c,d} = (\psi_c - \psi_d)\) for \(c < d\). Let \(r_c\) be any number in \([0,1]\). The difference between any \(r_c\) and \(r_d\) cannot exceed 1, so \(\delta_{c,d} = (\psi_c - \psi_d) \geq (r_d - r_c)\delta_{c,d}\). Let \(0 < \epsilon_A < \epsilon_A^U = \min\{\delta_{c,d} : c, d \in \{1, 2, 3\}, c < d\}\). Then we have \((\psi_c - \psi_d) > (r_d - r_c)\epsilon_A\), so \(\psi_c + r_c\epsilon_A > \psi_d + r_d\epsilon_A\). See Figure 1.

![Figure 1: Graphical depiction of the proof of Lemma 4.1](image)

Any change of the virtual valuations by \(\epsilon_A\) preserves the ordering of virtual valuations.

---

**Algorithm 3 \(\epsilon_A\)Finder**

1: \(\lambda \leftarrow \cup_{i \in N}\{\psi_i(z_{i,k}) : \forall z_{i,k} \in T_i\}\) \hspace{1cm} \(\triangleright\) All unique virtual valuations
2: if \(|\lambda| = 1\) then \hspace{1cm} \(\triangleright\) Everyone has the same virtual valuation
3: \(\epsilon_A = 1\) \hspace{1cm} \(\triangleright\) This satisfies \(0 < \epsilon_A < \epsilon_A^U\)
4: else
5: \(\epsilon_A^U \leftarrow \infty\)
6: Let \(\Lambda\) be an ascending sequence of virtual valuations in \(\lambda\)
7: for all \(\psi_k, \psi_{k+1} \in \Lambda\) do
8: \(\epsilon_A^U \leftarrow \min\{\epsilon_A^U, (\psi_{k+1} - \psi_k)\}\)
9: end for
10: end if
11: \(\epsilon_A \leftarrow \epsilon_A^U / 2\)
12: return \(\epsilon_A\)

A procedure for finding an \(\epsilon_A\) is given in Algorithm \[3\]. Constructing \(\lambda\) takes \(O(nR)\) time. Constructing \(\Lambda\) by creating a sorted list takes \(O(nR\log(nR))\) time. Therefore, the complexity of Algorithm \[3\] is \(O(nR\log(nR))\).

**Lemma 4.2.** There exists an \(\epsilon_B \in \mathbb{R}_{>0}\) such that for all \(r_{i,k} \in [0,1]\), if \(\psi_i(z_{i,k}) \geq 0\), then \(\psi_i(z_{i,k}) + r_{i,k}\epsilon_B \geq 0\), and if \(\psi_i(z_{i,k}) < 0\), then \(\psi_i(z_{i,k}) + r_{i,k}\epsilon_B < 0\), for all \(i, j \in N\), \(1 \leq k \leq |T_i|\), \(1 \leq \ell \leq |T_j|\).

**Proof.** Let \(\epsilon_B^U\) be the minimum of the absolute value of the set of all non-zero virtual valuations: \(\epsilon_B^U = \min\{|\psi_i(z_{i,k})| : \psi_i(z_{i,k}) \neq 0, \forall i \in N, \forall z_{i,k} \in T_i\}\). Let \(0 < \epsilon_B < \epsilon_B^U\). Any virtual valuation
\[ \psi_i(z_{i,k}) \geq 0 \] can have \( \epsilon_B \) added to it and remain non-negative. Similarly, any virtual valuation \( \psi_i(z_{i,k}) < 0 \) can have \( \epsilon_B \) added to it and remain negative, since \( \epsilon_B < |\psi_i(z_{i,k})| \). See Figure 2.

![Graphical depiction of the proof of Lemma 4.2](image)

Figure 2: Graphical depiction of the proof of Lemma 4.2. Any change of the virtual valuations by \( \epsilon_B \) does not change whether virtual valuations are negative or not.

Algorithm 4 \( \epsilon_B \)Finder

1: \( \chi \leftarrow \cup_{i \in N} \{ \psi_i(z_{i,k}) : \forall z_{i,k} \in T_i \} \setminus \{0\} \) \> Set of non-zero virtual valuations of all bidders
2: \( \epsilon^U_B \leftarrow 1 \)
3: for all \( \psi \in \chi \) do
4: \( \epsilon_B \leftarrow \min\{\epsilon^U_B, |\psi|\} \)
5: end for
6: \( \epsilon_B \leftarrow \epsilon^U_B / 2 \) \> This satisfies \( 0 < \epsilon_B < \epsilon^U_B \)
7: return \( \epsilon_B \)

A procedure for finding an \( \epsilon_B \) is given in Algorithm 4. The complexity of Algorithm 4 is \( O(nK) \), as constructing \( \chi \) takes \( O(nK) \) time.

We say that an \( \epsilon \) is a valid \( \epsilon \) if it satisfies the properties named in Lemmas 4.1 and 4.2 (and restated in Theorem 4.3). The existence of a valid \( \epsilon \) is given by the following theorem:

**Theorem 4.3.** There exists a valid \( \epsilon \in \mathbb{R}_{>0} \) such that for all \( r_{i,k} \in [0,1] \), the following properties hold:

- if \( \psi_i(z_{i,k}) > \psi_j(z_{j,\ell}) \), then \( \psi_i(z_{i,k}) + r_{i,k} \epsilon > \psi_j(z_{j,\ell}) + r_{j,\ell} \epsilon \),
- if \( \psi_i(z_{i,k}) \geq 0 \), then \( \psi_i(z_{i,k}) + r_{i,k} \epsilon \geq 0 \), and
- if \( \psi_i(z_{i,k}) < 0 \), then \( \psi_i(z_{i,k}) + r_{i,k} \epsilon < 0 \),

for all \( i, j \in N, 1 \leq k \leq |T_i|, 1 \leq \ell \leq |T_j| \).

**Proof.** Lemma 4.1 shows the existence of an \( \epsilon_A \) which satisfies the first property, and Lemma 4.2 shows the existence of an \( \epsilon_B \) which satisfies the latter properties. These values are not unique: for any \( \epsilon_A, \epsilon_B, \) we can construct an \( \epsilon_A' < \epsilon_A \) and \( \epsilon_B' < \epsilon_B \). Thus, the minimum of \( \epsilon_A \) and \( \epsilon_B \) satisfies all three properties of the theorem.

Algorithm 5 \( \epsilon \)Finder

1: \( \epsilon_A \leftarrow \epsilon_A \)Finder()
2: \( \epsilon_B \leftarrow \epsilon_B \)Finder()
3: \( \epsilon \leftarrow \min\{\epsilon_A, \epsilon_B\} \)
4: return \( \epsilon \)

A procedure for finding a valid \( \epsilon \) is given in Algorithm 5. The complexity of Algorithm 5 is \( O(nK\log(nK)) \), because this is the complexity of Algorithm 4 which is larger than the complexity of Algorithm 3, \( O(nK) \).

Theorem 4.3 tells us that unique virtual valuations may be changed without affecting their ordering among other virtual valuations, and non-negative (negative) virtual valuations can likewise
be changed and remain non-negative (negative). Thus, perturbing a unique virtual valuation will not alter its corresponding allocation.

Even more interesting, any tied virtual valuations which are perturbed will no longer be so, provided that changes to virtual valuations are unique. We use this observation to our advantage in order to compute total expected revenue in polynomial time.

Let the perturbed virtual valuation function \( \tilde{\psi}_i : T_i \rightarrow \mathbb{R} \) be defined as

\[
\tilde{\psi}_i(z_{i,k}) = \psi_i(z_{i,k}) + r_{i,k} \epsilon, \quad \forall z_{i,k} \in T_i,
\]

where all \( r_{i,k} \) variables are drawn independently from a continuous uniform distribution. Since all \( r_{i,k} \) variables are being drawn from a continuous distribution, the probability that \( r_{i,k} = r_{j,k} \) is 0, so each bidder’s perturbed virtual valuations should be unique.

**Remark 4.4.** Due to machine precision issues, it may be the case that there are non-unique \( r_{i,k} \) terms in an actual implementation of the perturbed virtual valuation function. In such a setting, a check may be implemented to see if this is the case. A new set of \( r_{i,k} \) values may be drawn again if non-uniqueness is observed. Drawing \( O(n \mathbb{R}) \) random numbers and checking for uniqueness is \( O(n \mathbb{R}) \), so this should not greatly affect runtime. (One may check for uniqueness by inserting each \( r_{i,k} \) into a hash table, keeping track of the number of times each \( r_{i,k} \) is seen.)

## 5 Revenue Maximization in Polynomial Time

We now show that drawing \( r_{i,k} \) from a \( U(0, 1) \) distribution is akin to picking a winner at random when there are ties. This means that we can compute total expected revenue without any error despite using perturbed virtual valuations when determining allocations.

**Theorem 5.1.** Total expected revenue can be computed in polynomial time using perturbed virtual valuation functions.

**Proof.** For any valuation vector \( v \in T \), Algorithm 2 will allocate only to bidders with the highest non-negative virtual valuation. As defined earlier, let \( w(v) \) be the set of bidders with the highest virtual valuation. Suppose instead of virtual valuations, we used perturbed virtual valuations. Let \( \tilde{w}(v) \) be the set of bidders with the highest perturbed virtual valuations that have met their reserve price:

\[
\tilde{w}(v) = \left\{ i : \tilde{\psi}_i(v_i) \geq 0, \tilde{\psi}_i(v_i) \geq \max_{j \in N \setminus i} \tilde{\psi}_j(v_j), \forall i \in N \right\}.
\]

Using a valid \( \epsilon \) guarantees that the intersection of \( w(v) \) and \( \tilde{w}(v) \) is nonempty. If there are no ties, then \( w(v) = \tilde{w}(v) \).

The interesting case is when there are ties. Since all perturbed virtual valuations are unique, \(|w(v) \cap \tilde{w}(v)| = 1\), and the unique bidder \( i^* \in w(v) \cap \tilde{w}(v) \) contributes \( \tilde{\psi}_{i^*}(v_{i^*}) \) to the total expected virtual surplus. The probability that \( i \in \tilde{w}(v) \) is allocated depends on the perturbations. Since perturbations are drawn independently and uniformly at random, the \( r_{i,k} \) values act as tie-breaking rules, where the probability that any \( j \in w(v) \) wins is uniform over the cardinality of \( w(v) \), just as in Algorithm 1. The maximum virtual surplus attained from any convex combination of winners in \( w(v) \) where \( \sum_{j \in w(v)} x_j(v_j, v_{-j}) = 1 \) is \( \sum_{j \in w(v)} \tilde{\psi}_j(v_j) x_j(v_j, v_{-j}) \), which is the outcome of Algorithm 2. In Algorithm 2, the virtual surplus given by \( v \) is \( \tilde{\psi}_{i^*}(v_{i^*}) \). Since \( \max_{j \in w(v)} \tilde{\psi}_j(v_j) = \tilde{\psi}_{i^*}(v_{i^*}) \), the contribution any \( v \in T \) has on total expected revenue is equivalent in both algorithms.

\[\square\]
Remark 5.2. Our choice of $\epsilon$ and $r_{i,k}$ ensures that virtual valuations are perturbed upwards. While it may be possible to perturb virtual valuations down with an appropriate choice of $\epsilon$ and values drawn from, say, a $U(-1,1)$ distribution, so that $\hat{\psi}_i(z_{i,k}) < \psi_i(z_{i,k})$ is possible, any virtual valuation equal to zero may become negative. Our choice of $\epsilon$ and $U(0,1)$ was made to avoid this possibility, so that a bidder with a zero virtual valuation may be allocated.

Remark 5.3. A distribution other than $U(0,1)$ may be used to generate perturbations, provided that the method described here is adapted to satisfy the conditions of Theorem 4.3, namely, that the tie-breaking rules employed should not alter the relative ordering of all virtual valuations is preserved, non-negative virtual values stay non-negative, and negative virtual values stay negative.

The analysis and remark given is not specific to virtual valuations, and can be applied to valuations as well. We can construct a perturbed valuation function, $\hat{v}_i : T \to \mathbb{R}$, where $\hat{v}_i(z_{i,k}) = z_{i,k} + r_{i,k}\epsilon$ for all $i \in N, \forall z_{i,k} \in T_i$, with a valid $\epsilon$ computed using valuations instead of virtual valuations, so we have the following corollary:

Corollary 5.4. Total expected surplus can be computed in polynomial time using perturbed valuation functions.

6 Interim Allocations and Payments

Let $v$ be a bidder profile where multiple bidders have the highest virtual valuation. Ordinarily, each bidder $i \in w(v)$ has probability $p = \frac{|w(v)|}{|v|}$ of being allocated: i.e., $x_i(v_i, v_{-i}) = p$. However, by using perturbed virtual valuations, allocations $\hat{x}_i(v_i, v_{-i})$ are either 0 with probability $1 - p$, or 1 with probability $p$. This means $\mathbb{E}[\hat{x}_i(v_i, v_{-i})] = 1(p) + 0(1 - p) = p$. While $x_i(v_i, v_{-i}) \neq \hat{x}_i(v_i, v_{-i})$, it holds that $x_i(v_i, v_{-i}) = \mathbb{E}[\hat{x}_i(v_i, v_{-i})]$.

Observe that $\hat{x}_i(v_i, v_{-i})$ is a Bernoulli random variable with mean $p$ and variance $p(1 - p)$. Assuming multiple runs of Algorithm 2, let $\hat{x}_i^d(v_i, v_{-i})$ be an allocation of the $d$th run. Let

$$y_{i,D}(v_i, v_{-i}) = \frac{\sum_{d=1}^{D} \hat{x}_i^d(v_i, v_{-i})}{D}$$

be the estimate of $\hat{x}_i(v_i, v_{-i})$ after $D$ observations of vector $(v_i, v_{-i})$. By Chebyshev’s inequality, for a fixed $a > 0$,

$$\Pr(|y_{i,D}(v_i, v_{-i}) - x_i(v_i, v_{-i})| \geq a) \leq \frac{p(1 - p)}{a^2 D^2}.$$  \hfill (19)

This suggests that for a fixed $v$, we can recover a good estimate of the allocation $x_i(v_i, v_{-i})$ obtained by Algorithm 1 with high probability after a few runs of Algorithm 2.

Interim allocations $\hat{x}_i(v_i)$ are dependent on the outcomes $x_i(v_i, v_{-i})$ for every $v_{-i} \in T_{-i}$. To obtain a good estimate of an interim allocation $\hat{x}_i(v_i)$ given by Algorithm 1, we can use a union bound to determine how many times we should run Algorithm 2

$$\sum_{v_{-i} \in T_{-i} : |w(v_i, v_{-i})| > 1} \Pr(|y_{i,D}(v_i, v_{-i}) - x_i(v_i, v_{-i})| \geq a) \leq \sum_{v_{-i} \in T_{-i} : |w(v_i, v_{-i})| > 1} \frac{|w(v_i, v_{-i})|^{-1}(1 - |w(v_i, v_{-i})|^{-1})}{a^2 D^2}.$$  \hfill (21)

In words, Chebyshev’s inequality tells us that the number of times we should run Algorithm 2 is directly proportional to the number of ties there are. When there are no ties, only one run is
sufficient to obtain the result given by Algorithm 1. An exponential number of ties would require an exponential number of runs to recover a good estimate of interim allocations given by Algorithm 1. With a good estimate of interim allocations, a good estimate of interim payments can then be computed by using Myerson’s payment formula, Equation (2).

Remark 6.1. As in [8], we assumed our problem is regular, so that \( \psi_i(z_{i,k+1}) > \psi_i(z_{i,k}) \) whenever \( z_{i,k+1} > z_{i,k} \). However, some works, such as [6], also use regularity to describe distributions, so that virtual valuations are not strictly increasing: i.e., \( \psi_i(z_{i,k+1}) \geq \psi_i(z_{i,k}) \) whenever \( z_{i,k+1} > z_{i,k} \). By adding small perturbations, we cannot guarantee that \( \tilde{\psi}_i(z_{i,k+1}) \geq \tilde{\psi}_i(z_{i,k}) \) whenever \( \psi_i(z_{i,k+1}) = \psi_i(z_{i,k}) \). In expectation, whenever \( \psi_i(z_{i,k+1}) = \psi_i(z_{i,k}) \), \( \mathbb{E}[\tilde{\psi}_i(z_{i,k+1})] = \mathbb{E}[\tilde{\psi}_i(z_{i,k})] \), so averaging the result across multiple runs of Algorithm 2 preserves virtual valuation monotonicity, and hence allocation monotonicity, in expectation.

7 Experiments

We close this paper by exploring the implications Chebyshev’s inequality has on calculating accurate interim allocations. To do so, we designed experiments which enabled us to empirically observe how an interim allocation generated by multiple calls to the Optimized Pointwise Maximizer, Algorithm 2, behaves. We measure how quickly, in practice, the Optimized Pointwise Maximizer finds a result close to that of the Pointwise Maximizer, Algorithm 1.

Specifically, we implemented Algorithm 1, the Pointwise Maximizer (PM), as well as Algorithm 2, the Optimized Pointwise Maximizer (OPM), the latter of which we ran multiple times, as described in Section 6. Additionally, we implemented a Monte Carlo method (MC), in which we sample valuation vectors to estimate the interim allocations. We provide results for the MC method because it is, like the OPM method, a fast, simple, intuitive way of estimating interim allocations and total expected revenue.

We use the PM method to compute desired interim allocations. We compute the average difference (error) of the interim allocations produced by OPM and MC versus the reference PM interim allocations,

\[
\frac{\sum_{i \in N} \sum_{v_i \in T_i} |\hat{y}_{i,D}(v_i) - \hat{x}_i(v_i)|}{\sum_{i \in N} |T_i|},
\]

where \( \hat{y}_{i,D}(v_i) \) corresponds to an interim allocation estimate from running a method \( D \) times.

The number of times we ran a method, \( D \), was based on the time it took the PM method to run, \( T_n \), which varies with the number of bidders, \( n \). For \( n \in \{2, 3, 4\} \), we ran the OPM and MC methods until their total running times matched \( T_n \). When using the Monte Carlo method, to generate one estimate of the interim allocations, we sampled \( 5 \times 10^3 \) valuation vectors. Thus, after \( D \) runs of the Monte Carlo method, we had sampled \( 5D \times 10^3 \) valuation vectors. Valuations were drawn from a uniform distribution, where \( T_i = \{1, 2, \ldots, 50\} \). All of the methods were implemented in Python, and ran on an Intel Core i5 4690 processor with 8 GB of memory.

The results of these experiment are reported in Figures 3, 4, and 5. The times it took to run the PM method are displayed as vertical lines. In each experiment, we found that in a fraction of the time it took to run the PM method, the OPM method provides very close estimates to the PM interim allocations, whereas MC does not. Thus, we see that not only is the OPM method easy to understand, but it is correct, in the sense that it produces symmetric interim allocations, as well.
8 Conclusion

We show how to solve for optimal revenue and surplus in the classic auction setting (one good, many bidders) in polynomial time without resorting to linear programming. Instead, our algorithm works much like Myerson’s classic optimal auction. First, we point out that expediting Myerson’s approach is easy when there are no ties. In the case of ties, we provide a method for altering virtual valuations slightly so that 1. ties are eliminated, and 2. any virtual valuation with a non-negative probability of winning maintains a very similar non-negative probability of winning. We then show how to use these modified virtual valuations to compute total expected revenue (or surplus, if applied to valuations) in polynomial time. Finally, we show how such a procedure can be used to obtain estimates of interim allocations and interim payments that are not serially dictatorial, thus preserving allocation probabilities. This analysis reveals that, while computing total expected revenue (or surplus) is always polynomial using our approach, recovering interim allocations and interim payments may take exponential time (in the case of exponentially-many

![Figure 3: The average interim allocation error after multiple runs of the Optimized Pointwise Maximizer and the Monte Carlo method for $n = 2$ using the uniform distribution. In the time it took to run the PM method (0.005 s), the OPM method generated allocations with an average error of 0.002, a small improvement from its initial average error of 0.005. The MC method, however, took longer than the PM method (0.006 s) to generate one estimate of interim allocations, with an average error of 0.15. Thus, fewer than 5,000 valuation vector samples must be used by the MC method to fit within the runtime of the PM method to get an error approximately two orders of magnitude worse than the OPM method.](image-url)
Figure 4: The average interim allocation error after multiple runs of the Optimized Pointwise Maximizer and the Monte Carlo method for $n = 3$ using the uniform distribution. The PM method took 0.27 s to finish. In the time it took to run the PM method, the MC method gave allocations with an average error between 0.08 and 0.05. The OPM method, however, gave allocations with an average error between 0.005 and 0.0004. By 0.05 s, the OPM method’s average allocation errors were less than 0.001. Both the MC and OPM methods average allocation errors improve quickly, but the OPM method always yielded results at least one order of magnitude better than the MC method.

Experimental results verify that the approach we describe here is not only simple; it is fast and yields highly accurate allocation probabilities (and correspondingly, expected payments). The experimental results also indicate a tremendous improvement in accuracy over Monte Carlo methods that attempt to estimate interim allocations by sampling valuation vectors. In the future, we would like to investigate how our analysis can be extended to other single-parameter settings, such as knapsack auctions and sponsored search.

9 Acknowledgments

This research was supported by NSF Grant #1217761.
Figure 5: The average interim allocation error after multiple runs of the Optimized Pointwise Maximizer and the Monte Carlo method for $n = 4$ using the uniform distribution. The PM method took 16 s to finish. In the time it took to run the PM method, the MC method gave allocations with an average error between 0.05 and 0.01. The OPM method, however, gave allocations with an average error between 0.006 and 0.00007. By 0.08 s, the OPM method’s average allocation errors were less than 0.001, and by 7.6 s, the OPM method’s average allocation errors were less than 0.0001. Both the MC and OPM methods average allocation errors improve quickly, but the OPM method always yielded results at least one order of magnitude better than the MC method.

References


