Accounting for Price Dependencies in Simultaneous Sealed-Bid Auctions

Brandon A. Mayer, Eric Sodomka, Amy Greenwald
Brown University
Providence, RI 02912 USA
{Brandon_Mayer,Eric_Sodomka,Amy_Greenwald}@brown.edu

Michael P. Wellman
University of Michigan
Computer Science & Engineering
Ann Arbor, MI 48109-2121 USA
wellman@umich.edu

Abstract

Current autonomous bidding strategies for complex auctions typically employ a two phased architecture: first, the agent predicts a distribution over good prices, and then the agent generates bids given those predictions, usually using a heuristic. For computational reasons, previous state-of-the-art methods assumed prices were independent across goods, and then bid based on marginal price distributions. However, prices for goods are typically dependent, especially for complements and substitutes. We develop computationally feasible methods for predicting joint price distributions, and employing such predictions in bidding strategies. We also demonstrate experimentally that the state-of-the-art heuristic for bidding in simultaneous second-price sealed-bid auctions is outdone by the analog of this same heuristic bidding with respect to joint instead of marginal price predictions.

1 Introduction

How should an agent bid in simultaneous auctions? One answer is: according to game-theoretic equilibrium. Although sensible in theory, this approach is rarely used in practice. First, a unique equilibrium cannot be guaranteed. Second, and perhaps more importantly, except for relatively simple settings (e.g., single-item auctions, or direct mechanisms with dominant strategies) (Krishna 2010), solved simultaneous-auction games are few and far between (Rabinovich et al. 2013).

An alternative approach—predominant, for example, in the annual Trading Agent Competition (TAC) (Wellman, Greenwald, and Stone 2007)—is a two-step process, whereby the agent first predicts auction prices, and second bids based on its predictions. Bidding strategies that base their decisions on price predictions are known as *price-prediction (PP) strategies*. *Optimizing* PP strategies produce bids that maximize expected utility given their predictions. Wellman *et al.* (2012) show that under standard assumptions, optimizing PP strategies are sufficient for bidding in simultaneous sealed-bid auctions, in the sense that any Bayes-Nash equilibrium can be expressed as a profile of optimizing PP strategies. This correspondence between decision-theoretic and game-theoretic solutions obtains only when price predictions are *s elf-confirming*, that is, when the realized prices are the predictions input to the PP strategies.

To our knowledge, all PP strategies studied in the literature that employ probabilistic price predictions do so on a per-good basis. That is, rather than construct a *joint* probability distribution over the prices of all goods, they predict *marginal* price distributions. This is done for computational reasons: both the prediction and the optimization problems are simpler when solved on a per-good basis. However, solution quality suffers when prices are not actually independent, which in general they are not—for example, in the presence of complementary and substitutable goods, as in the FCC spectrum auctions (Bykowsky, Cull, and Ledyard 2000) or the TAC market games (Collins, Ketter, and Sadeh 2010; Wellman, Greenwald, and Stone 2007). In this paper, we tackle the higher fidelity, but more computationally difficult problems of predicting joint price distributions, and then bidding based on these joint predictions.

We start in the next section by introducing definitions and notation. In Sec. 3, we develop computationally feasible methods for learning self-confirming price predictions using Gaussian mixture models. Sec. 4 describes bidding heuristics based on marginal and joint price predictions. The remaining sections describe the extensive set of experiments we ran to evaluate our methods, which taken together demonstrate the plausibility and effectiveness of accounting for price dependencies in simultaneous auctions.

2 Model

Consider a market in which n agents compete as bidders for the set of goods $\mathcal{X} = \{1, \ldots, m\}$, which are sold through simultaneous sealed-bid (one-shot) auctions. Each agent privately and simultaneously sends to the auctioneer a bid vector $\boldsymbol{b} = \langle b_1, \ldots, b_m \rangle \in \mathbb{R}^m_{\geq 0}$, where b_j is the agent's bid for good j. (A bid of zero is interpreted as a null bid.) The auctioneer thus receives a matrix of bids $\boldsymbol{B} \in \mathbb{R}^{n\times m}_{\geq 0}$, which it processes to determine the winner of each good and the payments assessed to each agent.

Let $\mathcal{B}$ be the set of possible $\boldsymbol{B}$ matrices. The auction outcomes are determined collectively by (1) an allocation rule $w : \mathcal{B} \to \{0, 1\}^{n\times m}$, which maps the agents' bids to a binary matrix specifying whether or not each agent won each good, and (2) a payment rule $c : \mathcal{B} \to \mathbb{R}^{n\times m}$, which maps the agents' bids to a matrix specifying the cost incurred by each agent for each good. In particular, $w^i(\boldsymbol{B})$ represents the set of goods allocated to agent i, and $c^i_j(\boldsymbol{B})$ is the payment assessed to agent i in the auction for good j.

Agent i's valuation function $v^i(w^i(\boldsymbol{B}))$ represents the value i attributes to its winnings. Since goods are allocated in separate auctions, payments are additive: $c^i(\boldsymbol{B}) = \sum_{j=1}^{m} c^i_j(\boldsymbol{B})$. Agent i's utility $u^i(\boldsymbol{B})$ equals its valuation less its payment: $u^i(\boldsymbol{B}) = v^i(w^i(\boldsymbol{B})) - c^i(\boldsymbol{B})$.

We focus on the standard allocation rule in which good j is allocated to the highest bidder for that good. We further assume that the winner of good j pays $\phi(b', b'')$, a function of the first-highest bid b' and second-highest bid b'' for that good, and the losers of good j do not pay. Now agent i's bid vector $\boldsymbol{b} \in \mathbb{R}^m_{\geq 0}$ together with the vector of highest other-agent bids $\boldsymbol{q} \in \mathbb{R}^m_{\geq 0}$ is sufficient to determine agent i's allocation and payment. The allocation rule reduces to $w^i(\boldsymbol{b}, \boldsymbol{q}) = \{j \mid b_j \geq q_j, b_j > 0\}$, and the payment rule to $c^i(\boldsymbol{b}, \boldsymbol{q}) = \sum_{j=1}^{m} c^i_j(\boldsymbol{b}, \boldsymbol{q})$, where $c^i_j(\boldsymbol{b}, \boldsymbol{q}) = \phi(b_j, q_j)$ if $b_j \geq q_j$ and $b_j > 0$, and 0 otherwise. Consequently, the utility function reduces to $u^i(\boldsymbol{b}, \boldsymbol{q}) = v^i(w^i(\boldsymbol{b}, \boldsymbol{q})) - c^i(\boldsymbol{b}, \boldsymbol{q})$.

At a high level, the bidder's problem is to determine bids that maximize utility. A *PP strategy* s is a function that takes as input predicted prices and a valuation function, and outputs a bid vector. More precisely, from the point of view of agent i, let $\boldsymbol{Q} = \langle Q_1, \ldots, Q_m \rangle$ denote a random m-vector (i.e., m random variables, one per auction), representing predicted highest other-agent bids, with probability density function (pdf) $f_{\boldsymbol{Q}}(\boldsymbol{q})$ and cumulative distribution function (cdf) $F_{\boldsymbol{Q}}(\boldsymbol{q})$: $F_{\boldsymbol{Q}}(q_1, \ldots, q_m) = f_{\boldsymbol{Q}}(Q_1 \leq q_1, \ldots, Q_m \leq q_m)$. Now $v^i(w^i(\boldsymbol{b}, \boldsymbol{Q}))$, $c^i(\boldsymbol{b}, \boldsymbol{Q})$, and $u^i(\boldsymbol{b}, \boldsymbol{Q}) = v^i(w^i(\boldsymbol{b}, \boldsymbol{Q})) - c^i(\boldsymbol{b}, \boldsymbol{Q})$ are random variables, and agent i's objective, when employing PP strategy $s(f_{\boldsymbol{Q}}, v^i)$, is to choose a bid vector $\boldsymbol{b} \in \mathbb{R}^m_{\geq 0}$ that maximizes its expected utility $\mathbb{E}[u^i(\boldsymbol{b}, \boldsymbol{Q})]$.

In the remainder of the paper, as we continue to reason from the point of view of just one agent, we drop the i superscripts, and write: $v(w(\boldsymbol{b}, \boldsymbol{Q}))$, $c(\boldsymbol{b}, \boldsymbol{Q})$, and $u(\boldsymbol{b}, \boldsymbol{Q})$. Furthermore, we abbreviate $v(w(\boldsymbol{b}, \boldsymbol{Q}))$ by $v(\boldsymbol{b}, \boldsymbol{Q})$. Lastly, we refer to sets of goods as *bundles*.

3 Learning Joint Price Predictions

Past work on learning price predictions has been concerned with learning marginal price predictions: making predictions on a per-good basis (Stone et al. 2003). Often, such price predictions are represented as histograms (Greenwald, Lee, and Naroditskiy 2009; Wellman, Sodomka, and Greenwald 2012). Assuming m goods, each with l possible discretized prices, it take $O(ml)$ space to represent m marginal histograms. The corresponding joint histogram would take space $O(l^m)$. Consequently, for large m, it is not possible to accurately model a joint price probability distribution as a histogram. Instead, we represent a joint distribution as a mixture of multivariate Gaussians.

Gaussian Mixture Models

A *Gaussian mixture model* (GMM) represents a joint distribution of dimension m as a weighted sum of K Gaussian components. Each component $k \in \{1, \ldots, K\}$ is defined by its mean vector $\mu_k \in \mathbb{R}^m$ and covariance matrix $\Sigma_k \in \mathbb{R}^{m \times m}$. In addition, each component is weighted by a scalar γ_k. The vector $\theta = \langle \gamma_1, \ldots, \gamma_K, \mu_1, \ldots, \mu_K, \Sigma_1, \ldots, \Sigma_K \rangle$ specifies the free parameters of a GMM with K components.

Given a GMM, the probability density $p(y) = \sum_{k=1}^{K} \gamma_k\, p(y \mid \mu_k, \Sigma_k)$, for $y \in \mathbb{R}^m$. The *likelihood* of a data set $\mathcal{Y} = \{y_1, \ldots, y_{|\mathcal{Y}|}\}$ with respect to θ is then $L(\mathcal{Y} \mid \theta) = \prod_{i=1}^{|\mathcal{Y}|} p(y_i)$.

A standard algorithm for estimating the free parameters of a GMM from a dataset $\mathcal{Y}$ is *Expectation Maximization* ($\mathsf{EM}(\mathcal{Y}, K)$) (Dempster, Laird, and Rubin 1977), an iterative approach which guarantees that the likelihood of the data never decreases; however, the number of components K must be specified in advance.

Rather than guess K, we use the *Akaike Information Criterion* (AIC) (Akaike 1974) to drive model selection. That is, we learn models for various values of K, and then select the model that minimizes AIC score. The AIC score is a standard measure of goodness of fit (i.e., the likelihood of the data under the model) discounted by the number of parameters $\kappa(K, m)$ in the model: $AIC(\mathcal{Y}, \theta, K, m) = 2(\kappa(K, m) - \ln L(\mathcal{Y} \mid \theta))$. To specify the mean of a single Gaussian requires m parameters. Further, the full covariance matrix requires $\frac{m^2+m}{2}$ parameters. So, in total, the complexity of our model is $\kappa(K, m) = K\left(m + \frac{m^2+m}{2}\right)$.

Alg. 1 formalizes our approach to learning the free parameters of a GMM given a data set $\mathcal{Y}$ and some maximum number of components $\overline{K}$. Next, we discuss our method of generating the price data our GMMs represent.

```
ALGORITHM 1: AIC_GMM
   Input  : data set $\mathcal{Y}$, maximum number of
            components $\overline{K}$
   Output: GMM parameter estimates $\hat{\theta}$
1  for $K \leftarrow 1$ to $\overline{K}$ do
2      $\hat{\theta}^K \leftarrow \mathsf{EM}(\mathcal{Y}, K)$
3      $aic^K \leftarrow \mathsf{AIC}(\mathcal{Y}, \hat{\theta}^K, K, m)$
4  return $\hat{\theta}^{K^*}$ where $K^* \in \mathrm{argmin}_K\ aic^K$
```

Self-Confirming Price Predictions

There are many ways one might build probabilistic price predictions from data (Wellman et al. 2004). We employ *self-confirming price predictions* (SCPPs), originally introduced in the context of simultaneous ascending auctions (Wellman et al. 2008), and further evaluated in SimSPSB auctions (Wellman, Sodomka, and Greenwald 2012). We extend the algorithms employed in these prior works to operate over a joint price space.

Our SCPP search procedure (Alg. 2) is an iterative process takes as input an auction environment Γ, a price-prediction strategy PP, a learning algorithm $\mathsf{LearnJoint}$, an initial price prediction $f^0_{\boldsymbol{Q}}$, and several parameters that control the process. During iteration t, the algorithm simulates M instances of Γ, with all agents playing strategy PP with price prediction $f_{\boldsymbol{Q}}^{t-1}$. These simulations vary across sample valuation vectors. Given the ensuing data set $\mathcal{Y}$, a new price distribution $f_{\boldsymbol{Q}}$ is learned. If that new distribution is sufficiently close to the old, then the new distribution is returned and the procedure terminates. Otherwise, yet another price distribution is formed by combining the new and the old distributions in some way (specified by function g), and the process repeats. As this procedure is not guaranteed to converge, it is forcibly terminated after L_{SCPP} iterations.

ALGORITHM 2: SCPP Search

```
Input  : environment $\Gamma$, PP strategy PP, algorithm
         LearnJoint, price prediction $f_{\boldsymbol{Q}}^0$, and
         parameters $L_{\text{SCPP}}$, $M$, $N_{\text{KLS}}$, $g$, $\tau_{\text{SCPP}}$
Output: price prediction $f_{\boldsymbol{Q}}$
1 for $t \leftarrow 1$ to $L_{\text{SCPP}}$ do
2     for $u \leftarrow 1$ to $M$ do
3         $\boldsymbol{v} \leftarrow$ draw a sample valuation vector
4         $y_i \leftarrow$ outcome of simulating $\Gamma$, with each
          agent playing PP$(f_{\boldsymbol{Q}}^{t-1}, \boldsymbol{v}_i)$
5     $\mathcal{Y} \leftarrow \{y_1, \ldots, y_M\}$
6     $f_{\boldsymbol{Q}} \leftarrow$ LearnJoint$(\mathcal{Y})$
7     if $KLS_{N_{\text{KLS}}}(f_{\boldsymbol{Q}}, f_{\boldsymbol{Q}}^{t-1}) < \tau_{\text{SCPP}}$ then return $f_{\boldsymbol{Q}}$
8     else $f_{\boldsymbol{Q}}^t \leftarrow g(f_{\boldsymbol{Q}}, f_{\boldsymbol{Q}}^{t-1})$
9 return $f_{\boldsymbol{Q}}^L$
```

All that remains before our search procedure is fully specified is to explain how we decide when two probability distributions are approximately equal. A standard measure of similarity between probability distributions is KL-divergence (Hershey and Olsen 2007): $KL(p,q) = \int_{-\infty}^{\infty} p(x) \ln\left(\frac{p(x)}{q(x)}\right) dx$. The KL-divergence is not a true distance metric because it is not symmetric, so we employ this symmetric version: $KLS(p,q) = KL(p,q) + KL(q,p)$.

Whereas there is no closed-form for the KL-divergence between two GMMs, we can interpret KL-divergence as an expectation, and approximate its value using Monte Carlo sampling. Drawing N_{KLS} samples (twice), namely $\boldsymbol{q}^k \sim f_{\boldsymbol{Q}}$ and $\boldsymbol{r}^k \sim f'_{\boldsymbol{Q}}$, we approximate $KLS(f_{\boldsymbol{Q}}, f'_{\boldsymbol{Q}})$ by $KLS_{N_{\text{KLS}}}(f_{\boldsymbol{Q}}, f'_{\boldsymbol{Q}}) = \frac{1}{N_{\text{KLS}}} \sum_{k=1}^{N_{\text{KLS}}} \left(\ln\left(\frac{f_{\boldsymbol{Q}}(\boldsymbol{q}^k)}{f'_{\boldsymbol{Q}}(\boldsymbol{q}^k)}\right) + \ln\left(\frac{f'_{\boldsymbol{Q}}(\boldsymbol{r}^k)}{f_{\boldsymbol{Q}}(\boldsymbol{r}^k)}\right)\right)$.

4 Bidding wrt Joint Price Predictions

Having addressed the questions of how to represent and make price predictions, we move on to how to bid, given those predictions. Since bidding decisions depend pivotally on auction rules, we focus from here on on a particular mechanism: the second-price sealed bid (SPSB) auction.

A wide taxonomy of heuristics has been proposed for simultaneous SPSB (SimSPSB) bidding (Wellman, Greenwald, and Stone 2007). We discuss two classes, one based on *marginal value*, and the other on *local search*. The former covers a large fraction of strategies previously studied in the literature, including varieties that implicitly or explicitly address the expected utility maximization objective. The latter is a flexible and computationally efficient approach to explicit optimization. While not optimal in general, the approach was shown effective in recent work (Wellman, Sodomka, and Greenwald 2012), and is significantly extended here.

Marginal Value Heuristics

When goods interdepend, the value of any individual good is not well-defined. Only its *marginal value* (MV)—its value relative to a bundle of other goods—can be quantified.

Definition 1. Given a valuation v, an agent's marginal value $\mu(v, j, X)$ for good j with respect to a bundle of other goods X is given by: $\mu(v, j, X) \equiv v(X \cup \{j\}) - v(X)$.

In the case of simultaneous auctions, *a priori* there is no bundle of goods with which to assess the relative value of a good. There are, however, predicted prices at which bundles can be acquired, and it is possible to assess the value of a good relative to the value of acquiring bundles at a cost.

Given a valuation v and a vector of prices $\boldsymbol{q} = \langle q_1, \ldots, q_m \rangle$, let $\sigma(v, X, \boldsymbol{q})$ denote the utility earned by acquiring the bundle X: $\sigma(v, X, \boldsymbol{q}) \equiv v(X) - \sum_{j \in X} q_j$. Defining $\sigma^*(v, \boldsymbol{q}) = \max_X \sigma(v, X, \boldsymbol{q})$, we extend the concept of marginal value as follows.

Definition 2. Given a valuation v, an agent's *marginal value* $\mu(v, j, \boldsymbol{q})$ for good j with respect to price vector $\boldsymbol{q}$ is given by: $\mu(v, j, \boldsymbol{q}) = \sigma^*(v, \boldsymbol{q}[q_j \leftarrow 0]) - \sigma^*(v, \boldsymbol{q}[q_j \leftarrow \infty])$.

Here, $\sigma^*(v, \boldsymbol{q}[q_j \leftarrow 0])$ denotes the maximal utility at the given prices, assuming good j is available for free, whereas $\sigma^*(v, \boldsymbol{q}[q_j \leftarrow \infty])$ denotes the maximal utility at the given prices, assuming good j is unavailable. The difference is precisely the marginal value of good j relative to the possibility of acquiring the other available goods at their respective prices. Since calculating σ^* entails optimizing over bundles, for general valuations, the MV-based heuristics take time exponential in the number of goods m.

The StraightMV heuristic takes as input price predictions in the form of point estimates, one per good, and bids marginal values with respect to those estimates. The related heuristic StraightMU takes as input price predictions in the form of (marginal) distributions, one per good, collapses the distributions into point estimates, and bids marginal values with respect to those estimates. Distributional price information can be collapsed either by computing $\mathbb{E}[\boldsymbol{Q}]$ directly—the so-called *expected value method* (EVM) (Birge and Louveaux 1997)—or by computing a sample average (SA) of this expectation: $\sum_{k=1}^{N_{\text{Q}}} \boldsymbol{q}^k$, where $\boldsymbol{q}^k \sim f_{\boldsymbol{Q}}$. We denote by StraightMUa the EVM version of this heuristic ("a" stands for analytic), and by StraightMUN_{Q} the SA version that draws N_{Q} samples.

It turns out that the StraightMU heuristics do not behave any differently under joint distributions than they do under the corresponding marginals. The reason for this is that the expected price of good j depends only on the random variable Q_j, and not at all on the random vector Q_{-j}.

Heuristics in the average-marginal-utility family bid on good j an estimate of $\mathbb{E}[\mu(v, j, \boldsymbol{q})]$, j's expected marginal value. They compute these estimates by first drawing N_{EMV} samples $\boldsymbol{q}^k \sim f_{\boldsymbol{Q}}$, and then computing: $\sum_{k=1}^{N_{\text{EMV}}} \sigma^*(v, q_j \leftarrow 0, \boldsymbol{q}^k_{-j}) - \sigma^*(v, q_j \leftarrow \infty, \boldsymbol{q}^k_{-j})$. For example, AverageMU64 samples 64 times from marginal distributions. The J AverageMU128 heuristic samples 128 times from the full joint distribution. (In general, we prepend a heuristic name with the letter J if that heuristic was formerly studied with marginal price predictions as input, but we feed it joint price predictions.)

Local Search Heuristics

A local search heuristic employs a local search in pursuit of its bids. Starting from an initial bid vector $\boldsymbol{b} = \langle b_j, \boldsymbol{b}_{-j} \rangle$, (e.g., one proposed by another heuristic such as StraightMV), an LS heuristic updates b_j for each good j in turn, holding all other goods' bids fixed. Each of the following local search update rules defines a heuristic.

JointLocal sets b_j to the expected marginal value of good j, relative to the bundles of other goods it might win, given bid vector $\boldsymbol{b}_{-j}$:

$$b_j \leftarrow \mathbb{E}[v(w(\boldsymbol{b}, \boldsymbol{Q}) \cup \{j\})] - \mathbb{E}[v(w(\boldsymbol{b}, \boldsymbol{Q}) \setminus \{j\})].$$

CondMVLocal also sets b_j to the expected marginal value of j, but conditions on winning j:

$$b_j \leftarrow \mathbb{E}\left[v(w(\boldsymbol{b}, \boldsymbol{Q}) \cup \{j\}) - v(w(\boldsymbol{b}, \boldsymbol{Q}) \setminus \{j\}) \mid Q_j \leq b_j\right].$$

CondMVLocal reflects dependencies between the current bid for good j (i.e., before updating) and $\boldsymbol{b}_{-j}$, whereas the JointLocal update rule ignores the current bid for good j.

The MargLocal update rule is identical to JointLocal's, except that the expectation is computed with respect to the marginals, rather than the joint:

$$b_j \leftarrow \mathbb{E}[v(w(\boldsymbol{b}, \boldsymbol{Q}') \cup \{j\})] - \mathbb{E}[v(w(\boldsymbol{b}, \boldsymbol{Q}') \setminus \{j\})].$$

Local search assuming independence was introduced by Wellman *et al.* (2012), who found it to be the best heuristic for the environments studied.

Proposition 1. *When prices are independent across goods, JointLocal and CondMVLocal reduce to MargLocal, so all three proposed local search update rules are equivalent.*

These local search update rules can be implemented using Monte Carlo sampling. Given N_{B} samples $\boldsymbol{q}^k \sim f_{\boldsymbol{Q}}$, as a representation of $f_{\boldsymbol{Q}}$, the JointLocal update rule, for example, is approximated by $b_j \leftarrow \frac{1}{N_{\text{B}}} \sum_{k=1}^{N_{\text{B}}} \big(v(w(\boldsymbol{b}, \boldsymbol{q}^k) \cup \{j\}) - v(w(\boldsymbol{b}, \boldsymbol{q}^k) \setminus \{j\})\big)$.

Our local search procedure is outlined in Alg. 3. Given a set $S = \{\boldsymbol{q}^1, \ldots, \boldsymbol{q}^{N_{\text{B}}} \mid \boldsymbol{q}^k \sim f_{\boldsymbol{Q}}\}$, the bid for each good is updated in turn, according to a local search update (LSU) rule. The procedure terminates if the Euclidean distance between the current and the previous bid vectors falls below a specified tolerance τ_{LS}, or when the maximum number of iterations L_{LS} is reached. Given sample $\boldsymbol{q}^k$, the winnings $w(\boldsymbol{b}, \boldsymbol{q}^k)$ can be determined in time linear in the number of goods m. Therefore, the overall complexity of Alg. 3 is $O(N_{\text{B}} L_{\text{LS}} m^2)$.

MargLocal was previously studied under the assumption that prices are independent across goods, in which case the expected marginal value of good j is in fact an optimal bid in simultaneous SPSB auctions:

$b_j \in \arg\max_{b'_j} \mathbb{E}[v((b'_j, \boldsymbol{b}_{-j}), \boldsymbol{Q})]$ (Wellman, Sodomka, and Greenwald 2012). Further, MargLocal converges to a locally optimal expected utility—it is not possible to change the bid vector in one dimension only and increase expected utility. In contrast, neither JointLocal nor CondMVLocal are locally optimal in this sense.

ALGORITHM 3: Local Search

```
  Input : update rule LSU, samples $S$, valuation $v$,
          initial bid vector $\boldsymbol{b}^0$, parameters $L_{\text{LS}}$, $\tau_{\text{LS}}$
  Output: bid vector $\boldsymbol{b}$
1 for $l = 1$ to $L_{\text{LS}}$ do
2     for $j = 1$ to $m$ do
3         $b^l_j \leftarrow \text{LSU}(S, v, \boldsymbol{b}^{l-1})$
4     if $||\boldsymbol{b}^l - \boldsymbol{b}^{l-1}|| \leq \tau_{\text{LS}}$ then
5         return $\boldsymbol{b}^l$
6 return $\boldsymbol{b}^l$
```

5 Valuation Environments

Our experimental evaluation is conducted in a variety of simultaneous second-price sealed-bid (SimSPSB) auction environments, employing four distinct classes of distributions over valuations. We denote an environment by $\mathcal{C}[m, n]$, with $\mathcal{C} \in \{\text{S}, \text{U}, \text{L}_1, \text{H}\}$ designating the valuation class, and m and n the numbers of goods and agents, respectively.

Valuation classes S, U, and L_1 correspond to *scheduling valuations* (Reeves et al. 2005). Under scheduling valuations, each agent i has a job to complete that requires the use of a common resource for $\lambda_i \in \{1, \ldots, m\}$ blocks of time. Each good j represents a time block during which the common resource will be allocated to the winner of good j. We write $d_i(j)$ to denote the value agent i receives for completing its job by time j. The earlier an agent completes its job, the higher its value (i.e., $d_i(j) \geq d_i(j+1)$). If an agent does not complete its job, it receives no value.

Scheduling valuations are of interest because they exhibit some form of both complements and substitutes. For example, when $\lambda_i = 2$, goods 1 and 3 are complements for agent i, since i obtains no value from either good alone, but does obtain value if it procures them both. At the same time, if agent i wins good 3, then goods 1 and 2 are perfect substitutes, since all that affects i's value is the time block of the latest essential good.

The valuation classes S, U, and L_1 differ only in how they generate λ_i and $d_i(t)$ values. In valuation classes S and U, λ_i is drawn uniformly from the set $\{1, \ldots, m\}$, whereas in valuation class L_1, $\lambda_i = 1$ (making all goods substitutes). In all classes, the $d_i(t)$ are drawn uniformly from the set of integers $\{0, \ldots, 50\}$. In environments S and L_1, the $d_i(t)$ are sorted to ensure monotonicity in j, whereas in U, for consistency with prior literature, the monotonicity constraint is enforced by ironing (Reeves et al. 2005).

In addition to scheduling valuations, we also consider valuations with more extreme substitutability. In valuation class

H (Wellman et al. 2008), the goods are identical; agents obtain value based solely on the total number they procure. Specifically, an agent's marginal value for the first good is drawn uniformly from the set $\{0, \ldots, 127\}$, and its marginal value for the jth good it obtains is uniform between 0 and its marginal value for the $j-1$st good.

6 Optimization Experiments

While previous sections have outlined efficient price prediction methods, this section reports on experiments over a range of bidding heuristics and valuation environments, designed to evaluate how correlations present in self-confirming price predictions impact optimization quality and the expected benefit of accounting for dependencies.

Experimental Setup

Because computing an optimal bid vector is intractable, even for small numbers of goods, and because previous work established MargLocal to be the best known bidder for the SimSPSB environment, we focus our optimization experiments on MargLocal and its joint counterpart, JointLocal.

We also include in our test suite of bidding heuristics a standard optimization routine, the Down Hill Method[1] (DHM) (Nelder and Mead 1965). DHM is a function minimization technique that starts at an initial candidate, evaluates the function, and then moves to a new candidate, making its selection from a pre-determined set of possible moves. The function's value is then re-evaluated at the new candidate, and the process iterates until convergence, or until a maximum number of iterations is reached. We tailored DHM to bid in SimSPSB auctions by minimizing negative expected utility, given price predictions. More specifically, the PP strategy JointDownHill estimates expected utility using Monte Carlo sampling, sampling from the full joint and using the same set of samples in all iterations. The PP strategy MargDownHill does the same, but samples from the corresponding marginals.

We compare the optimization performance of these bidding heuristics across various SimSPSB environments from valuation classes S and L_1.

Our method for evaluating the performance of a bidding heuristic in a given environment is as follows: First, to control for prediction accuracy, we generate a single SCPP for each environment. Second, for each environment, we fix a set of 1,000 sample valuations. For each valuation, we use J StraightMU8 to generate an initial bid vector, which is then passed as input to each heuristic. Other specific parameters passed to the LS heuristics are listed in Table 2. We then estimate the expected utility of each output bid vector using 10,000 sample price vectors.

Using the parameters listed in Table 1, we derived SCPPs in the form of GMMs using Alg. 2.[2]

We then measured the degree to which prices are correlated in each environment by estimating *total correlation* and observing *sample correlation* matrices in the corresponding SCPP. These two metrics are explained presently.

Parameter	Meaning	Value
PP	PP strategy	J StraightMUa
LearnJoint	learning algorithm	AIC_GMM
$\overline{K}$	maximum # of components	20
$f^0_{\boldsymbol{Q}}$	initial prediction	$U[0, \bar{V}]^m$
L_{SCPP}	maximum # of iterations	100
M	# of auctions simulated per iteration	10000
N_{KLS}	# of samples used to compute KLS	10000
g	combining old and new predictions	$g(f^{new}_{\boldsymbol{Q}}, f^{old}_{\boldsymbol{Q}}) = f^{new}_{\boldsymbol{Q}}$
τ_{SCPP}	KLS threshold (distribution)	0.001

Table 1: Parameter settings for SCPP search.

Parameter	Meaning	Value
τ_{LS}	convergence tolerance	1×10^{-5}
L_{LS}	maximum # of iterations	100
N_{B}	# of samples used to estimate bid	10000

Table 2: Parameter settings for local search.

Total correlation (TC) is a multivariate extension of mutual information, defined as $KL(f_{\boldsymbol{Q}}, f_{\boldsymbol{Q}'})$. This quantity is zero when the joint distribution is equivalent to the product of the marginals, and increases as $Q_1, \ldots, Q_m$ become more and more correlated. Given a price prediction represented as a GMM, we approximate TC via Monte Carlo integration. That is, given N_{TC} samples $\boldsymbol{q}^k \sim f_{\boldsymbol{Q}}$, TC $\approx \frac{1}{N_{\text{TC}}} \sum_{k=1}^{N_{\text{TC}}} f_{\boldsymbol{Q}}(\boldsymbol{q}^k) \ln\left(\frac{f_{\boldsymbol{Q}}(\boldsymbol{q}^k)}{f_{\boldsymbol{Q}'}(\boldsymbol{q}^k)}\right)$.

For a given price prediction, the sample correlation (SC) matrix can reveal further insight into the structure of price dependencies. Drawing N_{SC} samples as above, the sample covariance matrix is computed as: $C = \frac{1}{N_{\text{SC}}} \sum_{k=1}^{N_{\text{SC}}} (\boldsymbol{q}^k - \bar{\boldsymbol{q}})^T (\boldsymbol{q}^k - \bar{\boldsymbol{q}})$, where $\bar{\boldsymbol{q}}$ is the vector of sample means. The elements of the correlation matrix, ρ_{ij}, are computed by normalizing the elements of C: $\rho_{ij} = c_{ij} / \sqrt{c_{ii} c_{jj}}$.

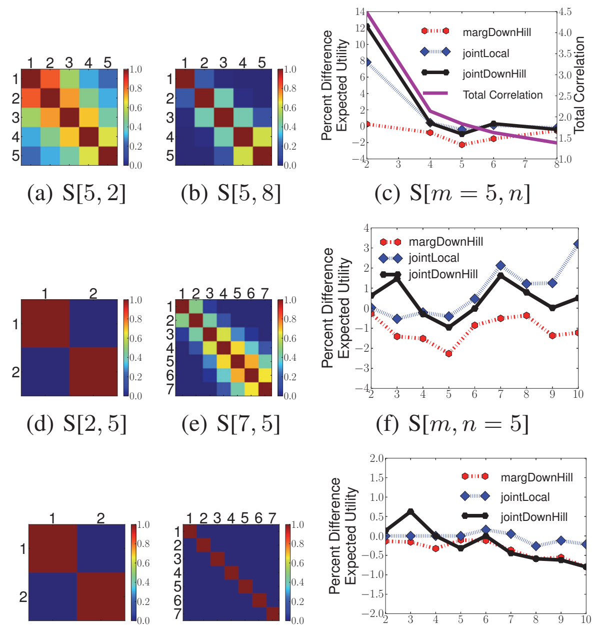
Experimental Results

Figs. 1(a) and 1(b) depict the sample correlation matrices for $\text{S}[5, 2]$ than $\text{S}[5, 8]$. Fixing the number of goods at 5, these figures suggest that prices are more strongly correlated in environments with fewer agents. (Dark red indicates strong correlation, whereas blue indicates weak correlation, or no correlation at all.) Likewise, Fig. 1(c) shows that two joint-exploiting heuristics (JointLocal and JointDownHill) outperform MargLocal by as much as 7.5% and 12%, respectively, when there are only two agents (i.e., strong price dependencies). On the other hand, when there are eight agents, and relatively weaker price dependencies, these two heuristics perform on par with MargLocal.

Also plotted in Fig. 1(c) is total correlation as it varies with n, the number of agents. This plot suggests that, given a fixed m, total correlation can identify values of n where independence assumptions will (or will not) significantly degrade heuristic performance. The knee at $\text{S}[5, 4]$ accurately segments this space of environments into two—those where the price independence assumption is particularly harmful

[1]We use the SciPy implementation (www.scipy.org).

[2]SCPP searches in this section use the scikit-learn implementation of EM (scikit-learn.org).



(a) S$[5, 2]$ (b) S$[5, 8]$ (c) S$[m = 5, n]$

(d) S$[2, 5]$ (e) S$[7, 5]$ (f) S$[m, n = 5]$

(g) $L_1[2, 5]$ (h) $L_1[7, 5]$ (i) $L_1[m, n = 5]$

Figure 1: Absolute value of sample correlation matrices for derived SCPPs in various environments, and the average percent change in expected utility of each heuristic as compared to MargLocal. For all sample correlation matrices, a dark red square at i, j indicates $\rho_{ij} \approx 1$, implying q_i and q_j are strongly correlated, whereas a blue square indicates $\rho_{ij} \approx 0$, implying q_i and q_j are uncorrelated.

($n < 4$), and those where it is not ($n \geq 4$).

Figs. 1(d), 1(e), 1(g), and 1(h) also depict sample correlation matrices, for the S$[2, 5]$, S$[7, 5]$, $L_1[2, 5]$, and $L_1[L, 5]$ environments, respectively. Here we fix the number of agents at five, and consider either two or seven goods.

When there are only two goods, the sample correlations are very weak off the diagonal in both of these environments. But in S$[7, 5]$, where there are seven goods, the sample correlation matrices exhibit stronger correlation factors off the diagonal. This observed increase in correlation, as the number of goods increases, is reflected in the relative expected utility estimates plotted in Fig. 1(f). When there are ten goods, JointLocal outperforms MargLocal by roughly 3%.

Also of interest in this plot is the divergence between JointLocal and the two downhill heuristics as the number of goods increases. Perhaps surprisingly, the downhill heuristics, which are derivatives of a generally high-performing optimization routine, are quickly surpassed by JointLocal, a provably suboptimal heuristic.

In the substitutes environment, the weak correlations seen in $L_1[2, 5]$ persist, even when there are seven goods. Intuitively, this makes sense: in any $L_1[m, n]$ environment, each agent requires a single good, and produces a bid for that good independently of all others. Correspondingly, the expected utilities of the various heuristics (including MargLocal) do not vary all that much with the number of goods, as seen in Fig. 1(i). Note, however, beginning at about $m = 5$, JointLocal starts to outperform the downhill heuristics. Again, we see that the downhill method struggles in higher-dimensional spaces. This result is yet another testament to the value of designing application-specific heuristics for complex optimization problems.

7 Empirical Game-Theoretic Analysis

To evaluate the effectiveness of bidding using joint price predictions, we conducted an extensive computational experiment, employing the methodology of empirical game-theoretic analysis (EGTA). This effort builds on a previous study in which we carried out an EGTA for SimSPSB over a comprehensive set of bidding strategies that employed price predictions, but assumed (incorrectly) that prices were independent across goods (Wellman, Sodomka, and Greenwald 2012). To the set of strategies considered previously, we added several that employ joint rather than marginal SCPPs. In all three environments studied—two with complement preferences and one with substitutes—we find that one or more of the new joint-exploiting strategies invades or completely overthrows the equilibrium of bidding strategies identified in the previous analysis.

Strategies and valuation environments

The original study evaluated a broad set of bidding strategies representative of prior literature, as well as the new (for that paper) MargLocal strategy. The mix included an array of MV strategies (see Sec. 4), using SC point-price predictions or marginally SC distributional-price predictions. It also included several varieties of BidEval, a meta-heuristic bidder that generates candidates using other (e.g., MV) strategies, and selects the one yielding greatest expected utility with respect to the input PP. The MargLocal heuristic rounds out the list of major strategy classes considered.[3] All of these strategies represent price distributions using m histograms, and assume the prices for respective goods are probabilistically independent. The procedure for deriving price predictions sought to find only marginally self-confirming PPs.

For the present study, we added to that baseline set three strategies that are direct joint-based counterparts of representative strategies from the original study:[4]

- J StraightMUa: A version of StraightMU (applies EVM to StraightMV, as described in Sec. 4), using the mean of the input PP.
- J AverageMU64: The strategy AverageMU, using 64 samples to estimate expected MV.
- J SCBidEvaluatorMixA_K16: A version of BidEval, employing 16 candidates generated by a mix of Aver-

[3]Space limits preclude a full recapitulation of the setup and results of the prior analysis. Whereas we provide here a complete description of our new experiments and findings, we rely on reference to the original study for detailed definitions of strategies and environments defined for that work.

[4]The strategy names in the report of the original study (Wellman, Sodomka, and Greenwald 2012) include a suffix "_HB" to indicate that the PP was derived using highest bids of other agents. In the current work this holds throughout, so we omit the suffix.

ageMU, StraightMU, and TargetMU. The bid candidates are evaluated using 32 samples from the input PP.

As observed in Sec. 4, EVM strategies like StraightMU behave no differently assuming independence or not. The implemented strategy J StraightMUa may nonetheless differ from StraightMUa, as the former's approximate SCPP was derived using the GMM representation, whereas the latter's used a histogram representation. In contrast, AverageMU and BidEval strategies behave differently under the independence assumption (unless of course the joint distribution actually exhibits independence).

We further added the joint-exploiting versions of the MargLocal strategy introduced in Sec. 4: JointLocal and CondMVLocal. We included a third LS variation as well, but this performed poorly and is not discussed further here. In our implementation, a run of a local search adjusts a bid vector until convergence, or 16 cycles through the goods, whichever comes first. Each strategy carries out four such local searches, initialized by a mix of different MV heuristics (as described for SCBidEvaluatorMixA_K16 above). It then employs the same 64 samples to estimate the expected utility of those four bid vectors, and selects the best one. If none yield positive expected utility, the strategy produces a null bid. With the exception of J StraightMUa, which uses the weighted component mean, all the new strategies sample from a GMM representation of a joint price prediction. Any negative prices drawn in these samples are treated as zeros.

We evaluated the six new strategies along with the baseline set in three environments: U$[5,5]$, U$[5,8]$, and H$[5,5]$. As described in Sec. 5, U valuations reflect complementary preferences, and H valuations perfect substitutes.

Deriving joint SCPPs

For each of the six new strategies in each of the three test environments, we derived a joint SCPP represented as a multivariate GMM. The procedure we employed follows the basic approach of Alg. 2, but differs in several particulars due to differences in the software environment employed for the EGTA portion of our study. We adopted a fixed number of GMM components for the PP representation ($K = 10$), and a fixed number of iterations of main search loop ($L = 50$, with no testing for earlier convergence). To damp potential oscillations in the search, the number of game samples M taken on iteration t is a decreasing function, $M_t = G(L - t + 1)/L$, where G (20,000 in our experiments) is the number of samples employed to fit a GMM. On each iteration we use the most recent G samples for this purpose.

Our GMM representation and EM algorithm for fitting the models to simulation data are implemented by the jMEF package[5] developed by Vincent Garcia and Frank Nielsen. To avoid degenerate GMM components, we represent each price point by a cloud of $m + m(m+1)/2$ points with small random perturbations.

[5]www.lix.polytechnique.fr/ nielsen/MEF

EGTA process and results

To describe our EGTA procedure, we need first to define some terminology. The *regret* $\epsilon(\boldsymbol{s})$ of a strategy profile $\boldsymbol{s} = (s_1, \ldots, s_n)$ (pure or mixed) is the maximum gain available to any player for deviating to an alternative strategy. A Nash equilibrium (NE) has regret zero. For an empirical game, payoffs are estimated by sample means, and a *regret bound* is defined with respect to the payoffs for strategy profiles that have estimates given the available simulation data. We say that a profile has *confirmed* regret ϵ iff ϵ is its regret bound and all possible deviations have been evaluated in simulation. We further define the *NE regret* of a strategy s with respect to a symmetric NE s^{NE} to be the loss due to playing that strategy relative to playing s^{NE}, when all other agents are playing s^{NE}. Note that since our empirical games are symmetric and finite, symmetric mixed equilibria are guaranteed to exist, and we focus attention on such equilibria in our analysis. To identify candidate equilibria in an empirical game, we first identify all *complete subgames*: subsets of strategies for which all profiles have been evaluated. We then apply replicator dynamics from several starting points to compute equilibria within these subgames, and test for overall equilibrium candidacy by calculating their regret bound with respect to the full strategy set.

To extend the earlier EGTA study, we started with the equilibrium found among baseline strategies, and simulated profiles for all one-player deviations to strategies in the set of new joint-exploiting strategies. We further simulated the six new strategies in self-play, and then executed the following two rules for generating new profile simulations, until quiescence:

1. Identify a symmetric mixed profile with *unconfirmed* regret bound less than a given threshold, and simulate all deviations from that profile.
2. Identify a subgame equilibrium s with positive regret bound, and simulate all profiles required to complete the subgame comprising the support of s plus its best known response among strategies outside that support.

The first rule ensures that at quiescence, any evaluated mixed profile either has a positive regret bound, or is a confirmed equilibrium. The second broadens exploration based on observed beneficial deviations.

Table 3 summarizes the extent of simulations covered cumulatively in this EGTA study, for the three subject environments. Each profile evaluated was simulated at least one million and typically two million times. As shown, the profiles evaluated represent only 0.08% to 1.51% of the possible profiles of their respective environments.

Environment	# Strategies	# Profiles	% Profiles	simulations (millions)
U$[5,5]$	36	5917	0.90	12573
U$[5,8]$	35	9554	0.08	18576
H$[5,5]$	40	16384	1.51	38364

Table 3: Numbers of strategies and profiles simulated for the environments addressed in our EGTA study.

As it happens, following the process above to extend the EGTA yielded exactly one confirmed equilibrium for each environment. In both $U[5,5]$ and $U[5,8]$, playing JointLocal is a pure-strategy NE. For the substitutes environment $H[5,5]$, the equilibrium is a mixture of five strategies: [BidXEvaluatorMix3_K16, 0.0625; SCLocalBidSearchS5K6, 0.0809; SCLocalBidSearch_K16Z, 0.1333; CondMVLocal, 0.6550; JointLocal, 0.0682]. While some of the independence-assuming strategies from the baseline set remain in equilibrium in $H[5,5]$, joint-exploiting strategies constitute over 70% of the support.

These equilibrium results confirm the value of exploiting joint distributions under both complements and substitutes, particularly given the breadth of coverage and strategy tuning employed in the predecessor EGTA study. Further evidence can be seen in the NE regret values presented in Table 4. In the complements environments, joint versions of AverageMU and BidEval outperform their marginal counterparts. The opposite is true for the substitutes environment, where these strategies are far from competitive anyway. Among the joint local search variants, we see that JointLocal and CondMVLocal are effective across the board.

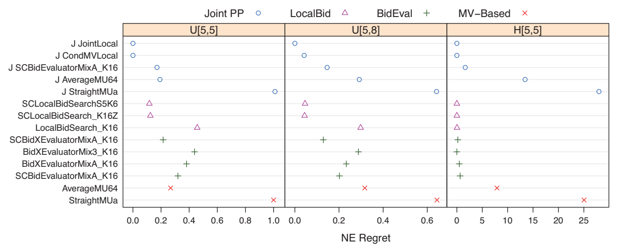


Table 4: NE regret for representative strategies.

8 Conclusion

In this study, we measure the benefits of accounting for price dependencies, compared to assuming prices are independent. These benefits can indeed be large, yet due to complexity concerns, prior research in trading strategy for simultaneous auctions has tended to assume independence.

Our main contribution is a set of techniques that enable trading agent designers to get beyond the independence assumption. We develop a general representation for joint price predictions based on Gaussian mixture models, and general computational techniques for constructing such predictions. We evaluate these techniques by generating self-confirming price predictions for a range of environments, and then measurin g the optimization quality of various bidding strategies that make decisions based on joint predictions. In an extensive empirical game-theoretic analysis comprehensively covering leading independence-assuming strategies from pr ior literature, we find that joint-exploiting strategies rise to the top. In particular, new local search heuristics that account for dependencies overthrow previous empirical equilibria to become reigning champion bidding strategies for these environments.

References

Akaike, H. 1974. A new look at the statistical model identification. *IEEE Transactions on Automatic Control* 19(6):716–723.

Birge, J., and Louveaux, F. 1997. *Introduction to Stochastic Programming*. Springer.

Bykowsky, M. M.; Cull, R. J.; and Ledyard, J. O. 2000. Mutually destructive bidding: The FCC auction design problem. *Journal of Regulatory Economics* 17:205–228.

Collins, J.; Ketter, W.; and Sadeh, N. 2010. Pushing the limits of rational agents: The trading agent competition for supply chain management. *AI Magazine* 31(2):63–80.

Dempster, A. P.; Laird, N. M.; and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B (Methodological)* 39(1):1–38.

Greenwald, A.; Lee, S. J.; and Naroditskiy, V. 2009. RoxyBot-06: Stochastic prediction and optimization in TAC travel. *Journal of Artificial Intelligence Research* 36:513–546.

Hershey, J. R., and Olsen, P. A. 2007. Approximating the Kullback Leibler divergence between Gaussian mixture models. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 4, 317–320.

Krishna, V. 2010. *Auction Theory*. Academic Press, second edition.

Nelder, J., and Mead, R. 1965. A simplex method for function minimization. *The Computer Journal* 7:308–313.

Rabinovich, Z.; Naroditskiy, V.; Gerding, E. H.; and Jennings, N. R. 2013. Computing pure Bayesian-Nash equilibria in games with finite actions and continuous types. *Artificial Intelligence* 195:106–139.

Reeves, D. M.; Wellman, M. P.; MacKie-Mason, J. K.; and Osepayshvili, A. 2005. Exploring bidding strategies for market-based scheduling. *Decision Support Systems* 39:67–85.

Stone, P.; Schapire, R. E.; Littman, M. L.; Csirik, J. A.; and McAllester, D. 2003. Decision-theoretic bidding based on learned density models in simultaneous, interacting auctions. *Journal of Artificial Intelligence Research* 19:209–242.

Wellman, M. P.; Reeves, D. M.; Lochner, K. M.; and Vorobeychik, Y. 2004. Price prediction in a trading agent competition. *Journal of Artificial Intelligence Research* 21:19–36.

Wellman, M. P.; Osepayshvili, A.; MacKie-Mason, J. K.; and Reeves, D. M. 2008. Bidding strategies for simultaneous ascending auctions. *B. E. Journal of Theoretical Economics (Topics)* 8(1).

Wellman, M. P.; Greenwald, A.; and Stone, P. 2007. *Autonomous Bidding Agents: Strategies and Lessons from the Trading Agent Competition*. MIT Press.

Wellman, M. P.; Sodomka, E.; and Greenwald, A. 2012. Self-confirming price-prediction strategies for simultaneous one-shot auctions. In *Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, 893–902.