Capstone Project: Optimizing Over Joint Price Predictions in Simultaneous Auctions

Leela Senthil Nathan

Supervisors: Amy Greenwald, Brandon Mayer

1 Introduction

What is the most effective bidding strategy in a simultaneous auction? Currently, it involves choosing the best bid via a price-prediction strategy. The price predictions for highest other-agent bids are represented using probability distributions, and optimizing over price-prediction strategies results in bids that maximize expected utility.

Previous work on learning price predictions has mostly been done on a per-good basis, i.e., using marginal price predictions. These price predictions can be further discretized and represented as histograms. However, in reality, price predictions reside in a continuous space—and the $O(k^m)$ space required to store a joint histogram for m goods with k possible discretized prices each is too unrealistic.

Working with marginal distributions is computationally favorable: both the prediction and optimization problems are much easier to solve when dealing with each good individually. However, this assumption that prices across goods are independent of each other need not necessarily be true—especially when working with complementary or substitutable goods—and this can hurt solution quality.

Thus, it is clear that continuous joint distributions are the most realistic representations of price predictions. My project involved using three algorithms—gradient ascent, marginal local search, and joint local search—to explore how effectively one can optimize over joint price predictions (for up to two goods) under three types of valuation functions to find the optimal bid vector.

2 Valuation Functions

Let $\mathbf{G}$ be the set of all goods. $v(\{\})$ is the value of obtaining neither good, $v(\{j\})$ is the value of obtaining only the jth good, and $v(\{j_1, j_2, ..., j_k\})$ is the value of obtaining goods j_1 through j_k.

We worked with three kinds of valuation functions, defining these in terms of any $\mathbf{X}, \mathbf{Y} \subseteq \mathbf{G}$:

1. Additive valuation: $v(\mathbf{X} \cup \mathbf{Y}) = v(\mathbf{X}) + v(\mathbf{Y})$
2. Sub-Additive valuation: $v(\mathbf{X} \cup \mathbf{Y}) \leq v(\mathbf{X}) + v(\mathbf{Y})$

3. Super-Additive valuation: $v(\mathbf{X} \cup \mathbf{Y}) \geq v(\mathbf{X}) + v(\mathbf{Y})$

3 Experimental Setup

We ran sets of 100 experiments at a time, working in the two-dimensional space $(0, 10) \times (0, 10)$. Bid distributions were defined using various combinations of Gaussian mixtures: a single bivariate Gaussian, a mixture of two bivariate Gaussians, and a mixture of five bivariate Gaussians). For each distribution, $\mu \sim \text{Unif}(0, 10) \times \text{Unif}(0, 10)$, i.e., each good's mean bid was sampled independently from the other's. Similarly, $\Sigma \sim \text{Wishart}(I, 4)$.

4 Experimental Results

4.1 Additive Valuation

According to a theorem we proved, in the event of additive valuation one can obtain the bid vector that maximizes joint expected utility by optimizing individually over each dimension—even when the goods are not independent of each other. Furthermore, by the second price auction theorem, an agent maximizes his expected utility for a good by bidding his true valuation for that good. Thus, given $v\{1, 2\} = v_1 + v_2$, $v\{1\} = v_1$, $v\{2\} = v_2$, the optimal bid must always be (v_1, v_2). All our algorithms converged to the optimal bid vector.

4.2 Superadditive and Subadditive Valuations

When working in a non-additive valuation domain, there is no theorem to guarantee what the optimal bid vector at convergence must be. Therefore, we focused on using experimental results to evaluate the performance of both algorithms when working with this kind of valuation function. Since optimizing over this valuation function is harder than in the additive valuation domain, we also compared performance with MATLAB's built-in gradient-based solver, *minunc*. For both types of valuations, joint local search outperformed the gradient-based algorithms (since they tended to get stuck in local maxima and minima).