Lab 8: Combinatorial Auctions, Part 1

Auction Rules

In today’s lab, you will build agents that bid in combinatorial auctions. At a very high level, these auctions work like this:

\[
\text{do} \{
\text{Get bids from each bidder}
\text{Check validity of bids}
\quad \text{If the set of valid bids is non-empty} \{ \text{Solve for tentative allocation} \text{Solve for tentative payment} \text{Send relevant information to each bidder} \}
\} \text{ while termination condition is not met}
\]

The first auction we will run today is a set of simultaneous second-price, sealed-bid auctions (SimSPSB) in which bidders place one bid per good. Pseudocode for this auction is as follows:

\[
\text{do} \{
\text{Get bids from each bidder, one per good}
\text{Keep only non-negative bids}
\quad \text{If the set of valid bids is non-empty} \{ \text{For each good} \{ \text{Allocate to a highest bidder} \text{Set the payment to the second highest bid} \} \text{Notify each bidder of their own allocation, and all prices} \}
\}
\]

This one allocation and these payments stand as the outcome of the auction.

The second auction we will run in today’s lab is a Simultaneous Multi-Round Ascending Auction (SMRA) (i.e., parallel English auctions, one per good), which operates like this:

\[
\text{do} \{
\text{Get bids from each bidder, one per good}
\text{Only keep bids on goods that exceed the good’s current price}
\text{If the set of valid bids is non-empty} \{ \text{For each good} \{ }
\]
Tentatively allocate to a highest bidder
Tentatively set the payment to the highest bid
}
Notify each bidder of their own allocation, and all prices
}
} until the set of valid bids is empty

The final allocation and final payments stand as the outcome of the auction.

Note that this auction allows jump bidding: i.e., placing bids that are much higher than the current price. A variant of this auction might truncate bids to be at most \( \epsilon \) above the current price. Rules like these which define what it means for a bid to be valid are called activity rules.

Agents’ Valuations

In today’s combinatorial auction simulations, as in the 2014 Canadian Spectrum auction, there will be 98 wireless spectrum licenses for sale. All agents’ valuations will be generated by the Spectrum Auction Test Suite (SATS). While SATS employs an internal representation capable of generating all \( 2^{98} \) valuations, your agent’s representation will consist of only a single XOR bid, comprised of a few atomic bids. This one XOR bid will fully characterize your agent’s valuation.

Implementation

In this lab, you will be writing a bidding agent that extends Lab8Agent. We will be using TradingPlatform as usual, so make sure that you add the TP jar to the classpath where your agent’s source code lives.

Agent

Your job is to implement onSimpleSealed (in the SimSPSB auction) and onSimpleOpenOutcry (in the SMRA auction).

An agent has one attribute, namely its valuation, myValuation, which is described by an XOR bid. More specifically, it is a Map<Set<FullType>, Double>, which maps a set of Tradeables (i.e., a subset of licenses) to a valuation.

As always, your agent’s valuation will be populated by the server before the game begins, and it is your job is to convert these valuations into bids.

Market

SimpleWrapper is the market that your agent will be interacting with in this lab. It exposes the following methods:
1. `BidBundle.BidderPrice getHighBid(FullType type)`
   a. This method takes as input a `TradeableType`, and returns a tuple consisting of of an agent ID and a Double. You can access the agent ID using `.AGENTID`, and the double using `.PRICE`. Note that the information revelation policy in both these games is such that `.AGENTID` is set to `null` when your agent does not win, but prices are always available. This method returns the reserve price (if any) in sealed-bid auctions and the current high bid in ascending auctions.

2. `void bid(Agent agent, Map<FullType, Double> bids)`
   a. This method takes as input an `agent` and a simple bid bundle, the latter of which maps a `Tradeable` to the price that your agent is willing to pay for it. You may omit from this `Map` any `Tradeable` that your agent is not bidding on. This method is used during sealed bid auctions (SimSPSB).

3. `void demandBid(Agent agent, Set<FullType> toBid)`
   a. This method takes as input an `agent` and a set of `Tradeables`, the latter of which your agent demands at the prices given by the `getHighBid` method of this `SimpleWrapper`. This method is used during open outcry auctions (SMRA).

Parameters

Here are the relevant parameter settings for today’s games:
- Your initial valuations will consist of only 3 atomic bids.
- You will have 2 seconds to submit your bids (i.e, the SimSPSB auction will last 2 seconds in total, and each round of the SMRA will last 2 seconds).
- The bid increment (€) is set to 20.

Exposure

Don’t worry if your agent achieves only negative scores in today’s games. Agents with combinatorial valuations can be subject to the exposure problem, where they win only a subset of the goods of interest to them, if the auction is not designed to mitigate exposure, which these week’s auctions were not. They were designed with simplicity in mind, only. Next week’s auctions will mitigate exposure, but they will also be more complex. Stay tuned!