CS404: Agent Based Systems

Matthew Sutton
University of Warwick
M.Sutton@Warwick.ac.uk

Abstract

This documentation details an approach to creating a competitive bot to function in an online private Auction House, subject to a specific set of rules. Two different games are considered: maximising total value, and acquiring a collection.

I. Introduction

Strategies for each game had to be modified to succeed in accordance with the Auction rules given. For the value game, Vickery bidding rules are used, where the highest bidder instead pays the second-highest bid. As opposed to the typical combinatorial auction, here currency has no value, and so leftover budget has no impact on utility. Additionally, the objective is no longer to increase wealth; instead, the format is competitive, making the goal solely to have the highest total value. Each round is Sealed-Bid, meaning bids are submitted by each party and evaluated simultaneously; as a player you have no view on the bids of other players during each round.

In the collection game, painting value is not of relevance. Here the auction becomes a race, with a goal painting combination given, and the winner being the first player achieving such a combination. This makes the artist the focus of each round, rather than the value.

Due to external circumstances I'm having to submit these solutions with limited testing, and so in places I will discuss further work that I believe has potential but have been unable to test.

II. Value Game

The setting for the value game is as follows. There are a fixed number of bots in the room to start with. Each bot begins with a budget of 1001, to use on a sequence of 200 paintings. At the end of these 200 rounds, the winner is the player with the highest painting value.

A. 2-Player Game

I approached the 2-player game with its own strategy, since it is a two player zero sum game, and so the value objective is fixed. In each round, one of the two players must win the painting, and so if value is viewed as a utility, one player winning the painting is equivalent to the other player losing it.

As a solution to this game, the first step is to calculate the total value in the game. We have access to the complete painting order, and so simply sum the values. Then our goal becomes $(Total\ Value + 1)/2$. If we exceed this value, we are guaranteed to win the game. With this in mind, the bidding strategy is driven by efficient bidding. In this context, we treat this as:

$$(CPV * Budget\ Remaining)/(Goal - Current\ Score) \quad (1)$$

where CPV is the Current Painting Value.

What this represents is taking our budget, dividing it by the amount of value required to win, then multiplying this by the value offered by the current painting. If we maintain this efficiency, we are guaranteed to win. An issue on top of this simple approach is the necessity of integer bids; we can't bid at exact efficiency all the time. Currently my solution simply rounds up for a more aggressive strategy, and this seems to be effective in most cases due to the Vickery bidding strategy. If an opposing player rounds down at any point, we win the painting, but pay the lower value. It would be interesting to attempt different rounding strategies, which would allow the player to exploit under/over bidding of the opposition.

B. N-Player Game

Strategising for games with more than 2 players is less simple. We can no longer play off a fixed objective value that guarantees victory, as it depends on the performance of other players. Similarly, I approached this using an Expected Utility function. In a typical Combinatorial Auction, your Utility is simply a combination of the value of the items you have won, and your budget. Since budget does not matter here, the initial approach is to optimise for the value of paintings you can win. However, the objective here is not to maximise your own Utility, but to beat the other bots in the game.

For a Utility function, I took inspiration from a paper titled 'Spiteful Bidding in Sealed-Bid Auctions' [1]. This led to a different approach to the Utility function:

$$U = \alpha_1 u_1 - \sum_{i=2}^{n} \alpha_i u_i \quad (2)$$

based on 2 vectors, $\{\alpha_1, \dots, \alpha_n\}$ and $\{u_1, \dots, u_n\}$. Each element in $\{u_1, \dots, u_n\}$ represents the expected Utility of the current painting for each of the n players. This is calculated in a similar fashion to (1); for each player we calculate the value they require to guarantee winning and apportion their remaining budget over this value objective. Winning value is given by:

$$ToWin = (Remaining + CurrentMax - Score)/2 \quad (3)$$

where Remaining is the value of the remaining paintings, CurrentMax is the highest current score in the game, and Score is the score of that individual play. If the value needed for a player to win is larger than the value remaining in the game, we use a ToWin of 1000, to avoid negative values. What this represents is that that twice of what a player needs to win must be greater than the value the current winning player gets if they win every painting. This

is since we can consider a player winning a painting equivalent to the other players losing it. As insight into our utility function, it is now clear that the structure is to maximise the difference between our value, and the value of other players. A standout issue is that we regard minimising the value of each opposition player equally at each round. Clearly this shouldn't be the case; when there are players that become irrelevant to the game due to being too far behind, we would rather aim to catch those ahead of us.

This is the goal of the scalar vector $\{\alpha_1, \dots, \alpha_n\}$. As outlined in [1], this represents how spiteful we aim to act towards each player and is updated every round. Key here is that $|\{\alpha_1, \dots, \alpha_n\}| = 1$, and $\alpha_i > 0$ for each player. Subsequently, setting $\alpha_1 = 1$ implies we play with complete self interest in a similar fashion to a typical auction strategy.

Finally we need some method for calculating what each value in $\{\alpha_1, \dots, \alpha_n\}$ should be. Here it is based on the value the player needs to win, and their remaining budget. The idea is that early in the game, players with high budgets have more influence, and later in the game players with high score have more influence. We care less about players that spend all their budget early, and players that preserve their budget until the end of the game. The simple strategy presented is to sum the ToWin values of every player and take α_i based on how close player i is to winning, proportional to the rest of the group. These spite coefficients are subsequently normalised. Again, the bid is rounded up to ensure we return an integer.

To explore this further, I would've liked to improve the accuracy of the spite coefficients by making them a combination of a player's current budget, score, and ToWin. Using an extra coefficient for each, reinforcement learning could be used to find the optimal way to adjust bids based on a player's position. Sadly I did not have access to extra bots in order to accurately learn what these coefficients should be; when using a suite of basic bots, bids normalised around 16 which gave consistent wins, but wouldn't be effective against more sophisticated bots.

Doing this would hopefully reduce some of the potential issues seen in testing, specifically that this approach struggles with very aggressive players. When a bot accrues a large value early, mostly depletes their budget, but still has a chance to win, then this strategy tends to spend a significant number of rounds playing aggressively towards the winning bot, at the risk of losing the game whilst other bots catch up. An alternative approach in larger games may be to invert the strategy, and have our bot play with large self interest in games where there are fewer leaders.

III. Collection Game

Approaching the collection game was a little simpler. In essence, the auction is a race, where rounds are played until a player obtains their collection. It makes little sense to approach this with a utility function, as value is gained through a number of specific paintings, making it more difficult to evaluate the impact of each painting on your chance of winning. Instead, it seems the obvious approach is a greedy algorithm[2], where the bot acts in complete self-interest, simply trying to win the game at as early a point as possible.

To core of this strategy is in calculating the earliest possible winning round, and the paintings required at that point. This is something that is done at the start of each round; taking our current collection, and assuming we win every subsequent painting, at which round is our goal met, and what does our collection look like at this point. Checking this is a simple loop, adding each upcoming painting to our collection one by one, each time comparing the paintings we would have the most of, against a descending sorted list of the target. As long as each target bucket is filled, we assume we win at this point, and check whether the current painting is necessary to this winning collection.

If the painting is not necessary, we simply bid a nominal amount. Though it may be sensible to bid 0, as we don't strictly want the painting in our collection, it allows for potential winning of cheap paintings, which may become part of a winning combination later in the game, particularly against multiple bots playing a similar greedy strategy.

If it is determined that the painting is necessary to the earliest possible win, then we bid a value based on out budget, and how many paintings are needed to win. Similar to the efficient bidding strategy given by (1), the idea is that if we over or under bid for the current painting, we strongly risk losing out on later paintings to those with higher budgets. As opposed to (1), the value of the painting isn't taken into account, as it is irrelevant to success. A nice caveat of this approach is that by default, at any point where the current painting would win the game, our bot bids the entirety of its budget.

In light testing this appears to work very well. Playing aggressively is clearly beneficial in the context, and the bot is able to pivot, aiming for different collections when current objective paintings are stolen. The main struggle seemed to be when these pivot collections were directly opposing to those that were previously the goal, making the paintings currently acquired inconsequential. I tested with a slightly deeper level of prioritisation, where our bot calculated a number of possible early win opportunities, and subsequently would lightly compete for paintings that would likely appear in these alternative winning cases, on the assumption that other bots would be competing for those in the earliest winning condition.

Issues arose with this softer strategy. Bidding more than a nominal amount on additional paintings had a large enough affect on budget that we became susceptible to the more aggressive strategies. Even in rooms where there were a number of bots playing aggressive strategies, often there would be one which managed to collect the paintings they needed. Although it was likely that if the auction were to continue, our bot would finish in second place, in this game there are no rewards for second place.

A second approach that succeeded in testing was to aim for the second earliest win, in cases where the painting order allowed for two distinct collections. When most bots are playing a greedy strategy, collecting uncontested paintings knowing they lead to an early collection was successful as long as the previous contested paintings were split between players.

Again, this faces many issues dependent on both the painting order, and the strategy used by other players. It requires the existence of a set of paintings that will both be

largely uncontested and completed early. Playing this strategy risks an aggressive bot winning the early collection, or pivoting into a similar collection that we cannot afford to contest. Overall it seems that playing a constantly aggressive strategy is the most consistently successful.

may use the solidus (/), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use a long dash rather than

IV. Conclusion

The Value game was approached using a Utility function with the goal of maximising the difference between our bot's value, and the value of the other relevant bots. With access to a larger testing suite, reinforcement learning could be used to tune the values in the vector $\{\alpha_1, \dots, \alpha_n\}$.

Collection games were approached using a greedy algorithm that looks for the earliest possible winning opportunity, and bids accordingly. Aggressive strategies will dominate here as the game ends as soon as any player owns a dominant collection. Again, with a diverse enough array of bots to play against, reinforcement learning could be used to place the probabilities on winning when playing for collections that are likely to be less contested; a greedy bot is predictable, and a better policy may be to have a tendency to look for alternate approaches.

References

[1] F. Brandt, T. Sandholm, Y. Shoham, "Spiteful Bidding in Sealed-Bid Auctions," 2007.

[2] E. Tardos, "Greedy Algorithm in General Combinatorial Auctions", 2012.