Strategy Documentation for the Bolt Auction

Team 4: Aditya Y., Teja D., Soumyasis G., Antony M.

Strategy:

We use a hybrid strategy that combines a Fixed and Reactive Strategy. A fixed strategy is one where the bot bids at fixed intervals whereas a reactive strategy is one where the bot bids in reaction to enemy bids.

Rationale:

When we do not consider the δ parameter, then we can mathematically prove that the reactive strategy is the best strategy. Since we have to consider the parameter δ, we realized that we could not do a purely reactive strategy and hence went for a hybrid reactive and fixed strategy.

Description of Strategy:

Initialization:

```python
def __init__(self):
    super().__init__()
    self.name = "Flash"
    self.e = {}             # Dictionary of last enemy bid for each auction_id
    self.tasks = {}         # Dictionary of asyncio tasks for each auction_id
    self.auctionEnded = {}  # Dictionary of auction_id that have ended
    self.multiplier = 1.126     # Enemy bid multipler. Set as 1 or 1.126
    self.start_bid = 0.2    # Starting Bid
    self.time_limit = 270   # Time limit on how long asyncio task should run for each auction
```

We initialized our starting bid at 0.2. This was empirically chosen, so that we don't have a slow start in our auction bids and so that we can win auctions that end quickly. Our objective is to perform a single step look-ahead on our own bids and our competing bidder bots, compare them and place the maximum as a bid. Hence, we select a value multiplier of 1.126, to estimate the possible next move of a competing bot and bid slightly more than that.

The distribution from which the duration of auction was sampled was provided to be exponential with mean 30 secs, hence we estimated empirically that the probability of auctions running more than 5 minutes (300 secs) is very less, hence limited our bid submissions to 270 secs to ensure that our bot would not suffer from timeout errors.

Comparison of bids:

```python
# If enemy has not bid yet
if auction_id not in self.e and curr_bid * 1.125 <= 1:
    curr_bid = curr_bid * 1.125
elif auction_id not in self.e:
    make_bid = False
# If Fixed bid is greater than reactive bid
elif curr_bid * 1.125 > self.e[auction_id] * self.multiplier and curr_bid * 1.125 <= 1:
    curr_bid = curr_bid * 1.125
# If Reactive bid is greater than fixed bid
elif self.e[auction_id] * self.multiplier >= curr_bid * 1.125 and self.e[auction_id] * self.multiplier <= 1:
    curr_bid = self.e[auction_id] * self.multiplier + 1e-5
else:
    make_bid = False
```

We compare our most recent bid in the variable `curr_bid` and the most recent highest competitive bid in `self.e[auction_id]`. We estimate the lookahead bids in respect to these bids by multiplying our own bid with

1.125 and enemy bid with 1.126, find the maximum and make that bid. We make sure that our bid doesn't cross the value of the item, 1.

Code Quality

We believe that our code quality is better than the other bots that utilize fixed strategies. The other bots simply placed a While loop in the Start auction or the Receive Bid functions of the bot. This could lead to timeout error and halts the simulation code as it is actually awaiting on any asynchronous call to the bot. Our bot launches an asynchronous task each time that the Start auction function of the bot is called and then returns immediately. The asynchronous task executes our bot's Run function which contains the While loop that executes the fixed strategy and some if-else conditional statements to implement the reactive strategy. We use dictionaries to coordinate this part with the incoming information of enemy bids and end of auctions.

Rationale behind performance

Where we Won

Consistency: Unlike most other reactive bots, our bot has consistent performance. Performance of reactive bots depends upon the other bots. The parameters like bid limit, starting bid and enemy bid multiplier are what differentiate the reactive bots. Reactive bots with higher values in these parameters will perform better than other reactive bots will lower values in these parameters, however they will get lesser profits the more the parameters are increased. The tradeoff between these two properties and the optimal value for the parameters is dependent on the enemy bots. Without knowledge of the enemy bots, choosing the optimal parameter values is luck based and hence the performance of the reactive bot cannot be guaranteed.

Fixed bots on the other hand provide consistency. The performance of a fixed bot can be ensured regardless of the parameters that the enemy bots use. While it might not be the strongest bot capable of beating every other bot, its performance can be ensured. In our trial runs before the final submission, using the other teams' warm up submission bot, our bot performed similarly to how it did in the final submission. Hence, we can say that our bot was consistent in its performance. Instead of using a pure fixed bot, we used a reactive bot as we made the assumption that there would be a high number of reactive bots and hence a level of reactivity is necessary in our bot.

Perks of having a look-ahead: Bots that are purely reactive may have to sometimes bid at least 1.125*previous_bid. Here to optimize, most bots bid only the bare minimum. Thereby, increasing the possible profit it can make. We noticed this and made a small amendment to our multiplier. It is 1.126 instead of 1.125. This combined with the fixed strategy of bidding at regular intervals makes it so that our bot acts one step ahead of other bots. We noticed this in some auctions where enemy bots are always bidding values slightly below our last bid even though they are bidding a few seconds after our last bid. We can notice the advantage this brings to the table in an auction against AGAPES (team 16 bot). In this auction, we won by bidding 0.69 and AGAPES bids 0.61 after the auction is over. We were those many steps ahead in the auction.

Where we Lost

As our bot is a hybrid of reactive and pure strategy, it has the same parameters as the reactive bots. Hence, our bot is at a disadvantage against other bots with higher values for these parameters. We decided to make the limit bid as 1. Bots that would have gone beyond 1 could have beaten our bot. Furthermore, our bot has an effective bid limit of around 0.92 as a result of our starting bid and our bid multiplier. So, bots that had a higher effective bid limit, as a result of their different start bids and multipliers, could have beaten our bot. Our bot hits it limit after roughly 1 minute. For example, in a match with Zoe that lasted 1 and a half minute our bot hit its effective limit of 0.92 within 1 minute, whereas Zoe had an effective limit of around 0.97.

Similarly with the starting bid and enemy bid multiplier, bots like the number one bot (Electro_Sapien), with starting bid of slightly higher than 0.2 (0.20171191), beat our bot in quick auctions. Additionally, bots with a higher multiplier enemy bid multiplier than our bot were able to stay a step ahead of us. For example, bots like Lupin2.0 with multipliers like

1.175 can bid ahead of us. Since they had such a high bid limit of 500, the issue of reaching bid limit faster due to higher enemy bid multiplier was not an issue.

Feedback

Possible Improvements: While the organizer has tried to make the code asynchronous and fast, it ended up with many bugs and in the end didn't help his or our cause. We were not able to run an entire tournament with all bots as we were getting many errors. Therefore, we finally resorted to 1vs1 auctions against other bots. This meant that we were not able to assess our bot fully before submitting it. We also believe that sharing warm up submissions put the bots that did well at an unfair disadvantage. We were lucky that since our bot did not perform well in the warm up submission, it did not garner the attention of other teams. The organizers should have informed the participants beforehand so that they can make necessary adjustments. I understand that the organizers were pressed for time, but the final logs could have been formatted better so that we would have been able to better and more easily analyze the auctions our bot took part in. Also, we felt that it would have been a better learning experience if the project were more related to the actual course content taught in class like the Use Case and Reading Assignments were.

Our experience with the Bolt Auction: At first, it seemed like the auction would generate random leaderboards especially because the auction could effectively stop at any moment. However, a few brainstorming sessions helped us to achieve that we could do consistently well despite all the so many non-deterministic components. Showing us that one could deterministically do well in such an auction.