StarGold15

Or Aharoni

June 2025

1 Agent Utilities

StarGold15 uses the following utilities in its negotiation strategies:

a custom convex aspiration class, defined as follows:

$$level(t) = \frac{y_0 - y_m}{x_m^2} * (t - x_m)^2 + y_m$$

so that:
y_0 is a hyper-parameter for the value of y the convex aspiration at $x = 0$.
x_m is a hyper-parameter for the x-axis value where the convex aspiration minimum point is at.
y_m is a hyper-parameter for the y-axis value of the the convex aspiration minimum point.

below you can see a picture of the custom aspiration function with $y_0 = 1, x_m = 0.95, y_m = 0.3$

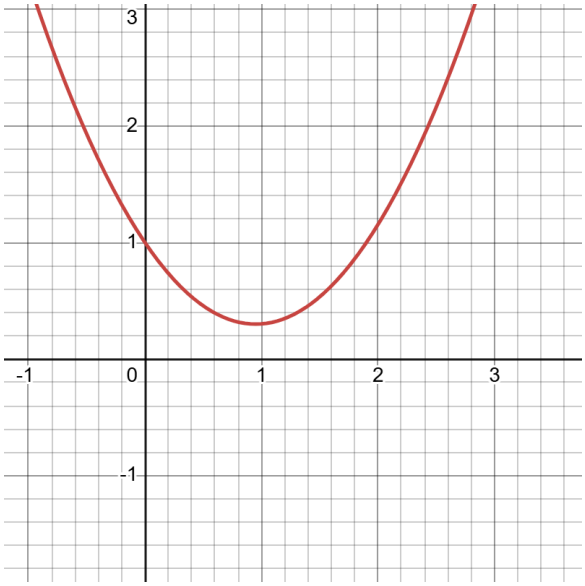


Figure 1: My custom convex aspiration function

ReduceThresholdBySubNegsLeft()
An assisting function I made which for each sub-negotiation returns a decreasing value between 0 to a certain parameter of maximum reduction between sub-negotiations (this number changes according to the utility function that we receive). for example, if that parameter value is 0.2, and we have 3 sub-negotiations, ReduceThresholdBySubNegsLeft() will return for sub-negotiation1 the value 0.0, for sub-negotiation2 it will return 0.1 and for sub-negotiation2 it will return 0.2.

2 Coordination

at the beginning of a sub-negotiation, StarGold15 creates a list which contains all the possible outcomes of the entire sub-negotiations (basically a tuple of length=`amount_of_sub_negotiations`). that list is created by Cartesian product of all opponents' outcome spaces, StarGold15 orders this list by the center utility supplied by the scenario (maximum, linear-sum, or lambda). After each agreement made by a sub-negotiation, the list is pruned to sequences consistent with the fixed prefix of deals, leaving a live pool of futures that are still reachable. All later decisions—both offers and acceptances—are evaluated against this shrinking pool, ensuring consistency with past commitments.

before each sub-negotiation, StarGold15 does a few checks and re-configures its utility calculations, assisting custom convex aspiration, as well as acceptance and offer strategies.
The most important check is:

what is the type of the utility function that it was provided with?

Max-type function

if its a Max-type function (meaning we get score based only on the best bid we had from all the sub-negotiations) the center agent adjusts the weights of our custom convex aspiration function to be quite high, this is because we need a single good bid, so we can be aggressive for all of our opponnent-edges until one of them yields a good enough bid.
StarGold15 uses the following formula for calculating the hyper-parameters of the respondpropose custom convex functions:
y-axis value of the minimum point is calculated as follows:

$$\texttt{propose\_curve\_new\_y\_axis} = 1.0 - ReduceThresholdBySubNegsLeft()$$

$$\texttt{respond\_curve\_new\_y\_axis} = 1.0 - ReduceThresholdBySubNegsLeft()$$

Here I set the max reduction by negotiation index of the ReduceThresholdBySubNegsLeft() to 0.4.
Note that I use the regular utility function that the agent was provided with in order to sort the possible bids.

linear combination function

If its a linear combination function (meaning that our result score is a linear formula of the utilities that we got in each sub-negotiation). In this case, StarGold15 re-initializes its custom convex aspiration function hyper parameters with the following formulas:
for the y-axis value of the minimum point, the following calculation is done:

$$\texttt{propose\_curve\_new\_y\_axis} = 0.45 + 0.5 * w_i - ReduceThresholdBySubNegsLeft()$$

$$\texttt{respond\_curve\_new\_y\_axis} = 0.45 + 0.5 * w_i - ReduceThresholdBySubNegsLeft()$$

where w_i is the weight of the current negotiation in the center utility function.
Here we set the max reduction by negotiation index of the ReduceThresholdBySubNegsLeft() to 0.2.
This technique is used to reduce StarGold15's aggressiveness near the ending rounds, while also making Sure StarGold15 considers how "important" this sub-negotiation is based on w_i.
Note that I use the regular utility function that the agent was provided with in order to sort the possible bids.

Lambda function
In the case that we cannot figure out the type of utility function that we got for this scenario, I aimed for a more general solution which involves re-calcuation the utility of each possible bid of the current sub-negotiation. I count all the outcomes that have the prefix of bids of the previous negotiations and then the current bid. I then calculate the average utility score of all those outcomes, and also normalize those outcomes amount to its relative size against the total amount of outcomes,
so we look at the average utility in the **future** for that bid if we took that bid, and also how "rare" that future is. and so the final formula of re-calculating the utility is as follows:

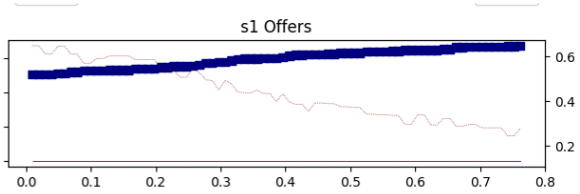
$$newUtility(bid) = avg_{possibleOutcomes} * 0.6 + AmountOfOutcomesNormalized(bid) * 0.4$$

note that $possibleOutcomes$ are all the outcomes that have the prefix of bids of the previous negotiations and then the current bid.
and $AmountOfOutcomesNormalized$ are the amount of those outcomes normalized to its size compared to all the possible outcomes.

* note that in order to not count outcomes that are too small, i limit the search space to outcomes that have a utility value above a certain threshold.

3 Bidding strategy

My bidding strategy is a lie. why is that?
after calcuting the utilities as specified in the coordination section, StarGold15 arranges its bids by their utilities, while removing bids that have less than the minimal value of y that was set on the custom convex aspiration function (this is in order to prevent offerring bids that has too little value for us).
And then the lie starts. many of the other agents uses some kind of opponent modeling, probably something like "the first bid is probably the best for the opponent" or "if the bid appears a lot, its probably good for the opponent". And so what StarGold15 does is that it flips its array of bids, so that the "worst bid" (remember that we filtered bids that are too little of value to us, so even the "worst bid" is actually really good) is first, and the best bid is last, and then StarGold adds again the best bid at the end.
for example if the best bid is 5 and then 4 3 2 1, we will have a bid list like this: 1 2 3 4 5 5.
then StarGold15 proposes each bid for the same number steps until the end of the negotiation. This way, StarGold15 fools its opponent into learning a false utility of the opponent, and make him think that we are conceding and that it should wait to get the deal that is the worst for us. while also making sure that even if the opponent accepts the very first (worst) offer that we proposed, we still end with a pretty good utility). here is an example of how our fake conceding looks:



4 Acceptance strategy

StarGold15 accepts a bid in conceding order, meaning that at the first few rounds, StarGold15 will only accepts bids if they are extremely close to the best bid (which was calculated as explained in the Coordination section), as the time of the negotiation progresses, StarGold15 will accept bids that are a bit further from the best bid by utility, this is done using the `respond_curve` that I wrote about earlier. the curve is called in each respond() function and its value (as shown on page 1) starts from 1 and slowly reduced to a certain minimal threshold (`respond_curve_new_y_axis`).
so StarGold15 accepts a bid at certain time since start of negotiation if:

$$ufun(bid) >= \texttt{best\_bid\_score} * \texttt{respond\_curve(relative\_negotiation\_time)}$$

5 Fail prevention and opponent modeling

StarGold also monitors its opponent offers thorughout the negotiations, it keeps each offer made by the opponent and creates a custom utility function of the opponent by the frequency of the offers. StarGold then uses this data for two failsafes for when the negotiation is almost at the end and no offer was accepted (this is to prevent walking away from a negotiation with nothing): for the respond part, StarGold calculates the nash equilibrium with bids that are above the threshold set by the `respond_curve` minimal point divided by 1.5, and will accept an offer if its above the utility of the bid that is the nash equilibrium.

for the propose failsafe, StarGold15 goes through all the offers made by the opponent, if one of the offers' value is close to the value of the best bid for StarGold by a low 0.3, StarGold15 will propose it as a compromise.

6 Other StarGold15 tricks

after each sub-negotiation, StarGold15 checks: "if from now on i will never propose an offer or accept an offer, how close i am to the best possible outcome for me?" and so StarGold15 does this check, and if the outcome of doing nothing form here on is close by 0.92 to the best outcome possible, StarGold15 will simply do nothing (returning None) and reduce the score of all the next edge agents that will attempt to negotiate with it. (Note that this is really useful is cases like a Max Utility Function).