1 Agent Design

We mainly focus on the changes of bids over time. The opponent agents start with highest bid in their preferences. As time goes by the opponent agents try not to change important issue values to get more utility. In this assumption if one issue value is no change in recent bids, this issue value provide more utility to the agent which gives the bid. We use same idea when considering history. If one issue value given after than another value, recent value is more preferred value for this agent. Therefore when we considering bid weights, we give more weight to recent issue values. To use this information we evaluate bid values in 2 way. First we consider real utility values for our agent. The second utility values come from evaluation of other agent bids. First all bid weights start from zero. After opponent agent gives a bid, we give weights to the issue values that comes inside with the given bid. We use time based weights when we give weight to issue values. We multiply elapsed time with constant then add this values to issue weight. After we calculate weights of issue values according to given time of value and frequency, we forecast the bid utilities for opponent agents according to our weights. For every agent we keep separate value map to understand each agent preferences. In order to achieve most utility bid for every agent, first we consider our utility space. We filter our utility space according to time threshold that is decreased by time. We take bids above time threshold in our utility space then we sort bids based on their utilities. As a next step we calculate these sorted bid's utilities for other agents using our weight map. We sum weights of issue values for each bid. This summation gives the utility of the opponent agent according to our assumptions. To find total utility of the bid for all agent in a negotiation we sum all agents' utility for the bid.

The second strategy we use in our agent is bid looping. At the start, we give Max utility bid as an offer 4 times and put these values inside an ArrayList. For the 5th offer we generate one new bid using our time-frequency strategy. After new bid created, we delete the first bid in the ArrayList and put the new bid to the end of the ArrayList. Again we have 4 bid in our ArrayList. Now we give our offer from the start of the ArrayList. After we give the all bids in the ArrayList as an offer from start to the end, we generate a new bid. After this step we repeat same process till the end of the negotiation. This strategy provides 2 main benefit. We give same bids with 5 times with same order. The opponents that are using frequency models increases weights of issue values in high utility bids because these bid are given 5 times. Using this strategy the opponent can forecast our agent preferences more easily. The second benefit of bid looping is that if the opponent agent use titandtat strategy, our agent will concede at first then go back to high utility bid. Because bids are sorted according to our bid utility in the ArrayList when we use bid looping, the opponent agent will read that move as our agent is conceding.

The third strategy is using history according to bid order. We give more weight to later bids' issue values. This strategy makes it easier to find out which values are indispensable for other agents. Using this information, we can increase the utility of the opponent agent by keeping our own utility fixed.

1.1 The Time Conceding Strategy

Our agent does not concede according to time. We use the time in our bidding strategy. As time goes on, our agent updates the range that we use to get the bids from our utility space. We divide time-space into 3 intervals based on remaining time ratio. These intervals are the remaining time ratio is greater than 0.5, the remaining time ratio is between 0.5 and 0.2 and the remaining time ratio below 0.2. For the case of the remaining time ratio greater than 0.5, we are using 0.85 + the remaining time over 1/e as a threshold value for generating bids. After we calculate the threshold for generating bids, we take all bids which have utility more than the threshold value. After collecting these bids in a pool, we sort them based on their utilities. As a next step, we forecast these bids' utilities using our time-frequency strategy. Lastly, we take the best bid in the pool which gives more utility considering our utility space and forecast calculations. Our agent uses the same strategy for other 2 cases. The only difference is between the cases is their constants values that are used for calculating threshold value. In case we use 0.65 instead of 0.95. In the last case, the constant value is 0.7.

1.2 Accepting Criteria

In the first three round phase, we give Max utility offers. If the opponent agents give max utility input in first 3 phase our agent directly accepts it otherwise gives a new offer. After the third round, our agents accept if the opponent agents' bid utilities are greater or equal than our agent next offer's utility, our agent will accept a bid. In the tournament results, we see that this approaches can end up with disagreement mostly if our agent not concedes. To solve this problem we change acceptance criteria. When %3 of the time remains to the end of the negotiation, our agent directly accepts the bid if the utility of the bid is higher than 0.65.

1.3 Deciding to Finish of negotiation

Our agent decides to finish the negotiation according to time and threshold. After %97 of the negotiation time passed, our agent makes a choice between 2 decision. If the utility of opponent agent' bid is greater than 0.65 which is our threshold value for finishing negotiation, our agent accepts the bid. If the utility of opponent agent' bid is smaller than 0.65, our agent ends the negotiation.

1.4 Opponent Modeling

Our agent holds the weights that we find using our time-frequency strategy in HashMap structure. Each opponent has own Hashmap. In the key part of HashMap structure, we hold issues. In value of Hashmap structure, we hold another Hashmap which contains the values of issues and their weights. To do looping we use ArrayList structure. Our agent updates the opponent models when new offer received.