Negotiation Agent

Intuition behind the design:
If an agent concedes too rapidly to reach a quick agreement, then that agent will be taken advantage of by agents who don't share their opponent's temporal preference. Further improvement can be achieved by learning the opponent's preference profile and then offering a bid likely to be accepted by the opponent without rapidly conceding.

Design of the Agent:
Somewhat similar to Boulware, the agent is hardheaded i.e., it will not concede until the very end. Using a concession function, it generates bids in a monotonic way, which resets to a random value after the dynamic concession limit is reached. In practice, this means that most of the time the agent will cycle through the same range of bids. Since the preferences of the opponent are not known, the agent tries to learn the opponent's preference profile. It chooses bids which it thinks are optimal for the opponent in case there are equivalent bids for itself.

General Strategy:
The agent uses a Boulware function during the negotiation, to calculate the utility P to which the agent is willing to concede at that moment. A concession function F determines the concession limit.

$$P = \text{MinUtility} + (1 - F(t)) \cdot (\text{MaxUtility} - \text{MinUtility})$$

$$F(t) = k + (1 - k)\left(\frac{\min(t, T)}{T}\right)^{\frac{1}{e}}$$

Here t is the current time, T is the total negotiation time, k is the initial concession and e is the concession rate coefficient. On discounted domains, the discount factor is ignored when it is greater than 0.8, otherwise the agent changes e to a value higher than 1. As on a scale from 0 to 1, time passes ahead of the discount value. The effect is that the agent suddenly changes strategy and concedes rapidly to prevent further loss of utility. The agent keeps a threshold MinUtility below which no offers are accepted. Bids that are generated are all in a narrow range deemed equivalent for itself. This range will increase monotonically towards the concession limit defined by P(t), which when reached causes the agent to reset the range to a random value greater than the concession limit. In case the learning function performs inadequately, the agent also remembers the best bid that the opponent has generated so far. The agent will give preference to this bid over the ones generated by the agent's own learning module, provided that the opponent's' best bid is above the current reservation value.

Learning Module:
Domains are assumed to contain multiple issues where each issue has multiple values. The utility for each value is assumed to be unknown. Furthermore the values are unordered. The utility for a particular bid b is denoted by Ub and is the weighted sum of each of the selected values in b.

$$U_b = \sum_{n=1}^{N} w_n \times v_{n,i}$$

Here w_n is the weight for issue n and $v_{n,i}$ is the utility of value i that is assigned by b to issue n. The goal of the learning module is to learn the weights and utility value for the opposing agent. To do this the agent makes two implicit assumptions regarding the opponent. Firstly, it assumes that the opponent restricts the bids it makes to a, possibly moving, limited utility range. Secondly, opponent prefers to explore the acceptable range rather than simply offering the same bid over and over again. Our learning function is a greedy reinforcement learning function, which updates the issue weights and value utilities of the preference profile after each bid. The update rule is split into 2 parts, one for the weights the other for the issue value utilities.

if $T == 1$ **then**
  Initialize preference profile
else
  set *index* to where $V(T) == V(T-1)$
  $W(index) = W(index) + \varepsilon$
  $W = W / \sum_{n=1}^{N} w_n$
  V(index)=V(index)+1
end if

The algorithm checks for issue values that the opponent has kept unchanged over all its offered bids. It then adds a value ε to each of those issue weights and the weights are normalized every round. The values get their utilities incremented each time they remain unchanged. Since these utilities are not required to be percentages, they only normalized to their maximum value per issue at the time the utility is calculated.
At each turn, we generate 100000 random bid and out of these bids, the bid whose utility is greater than our target utility and has maximum overlap with opponent's bid is selected and given. An upper bound is also set with utility 0.3 greater than target utility.
There are situations where our algorithm is not able to estimate the preference profile aptly. Especially when the opponent follows a non-monotone pattern of concession and bid offering, the learning module can lose track of the most important issues and issue values. This can also happen when the opponent has a utility profile which has multiple equally important issues.

Setting Parameters:
The parameters, k, minUtility and e, are set by experimenting on Rental House Domain. They can be easily learned for a new domain

Discounted Domains:
The discount factor undermines the overall strategy because the general strategy is to hold out as long as possible. The discount factor makes it attractive to reach an agreement as soon as possible. However the constraint is still that if an agent concedes too rapidly to reach a quick

agreement then that agent will be taken advantage of by agents who don't share their opponent's temporal preference. As mentioned earlier, The agent creates a virtual deadline at Tmax × δ where δ indicates the discount factor. Also a dynamic minimum utility threshold is calculated according to the following equation.

$$\text{DynamicMinUtility} = \text{MinUtility} - (\text{MaxUtility} - \text{MinUtility}) \times \delta$$

with the constraint that DynamicMinUtility cannot go below reservation value.

Conclusion:

The agent has a low computational complexity, this enables it to generate bids very rapidly. This celerity allows the agent to concede quickly when the time of the negotiating session is about to finish, whilst still thoroughly exploring the bid space. It uses a simple learning module and an optimal concession function to generate bids. The agent does not need a more sophisticated learning algorithm because it has ample time to explore the bid-space and offer any bid. An improvement can be done by estimating the opponent's rate of concession for discounted domains and switching strategies in accord.

Reference:

Thijs van Krimpen, Daphne Looije, and Siamak Hajizadeh: HardHeaded