HardHeaded

Thijs van Krimpen, Daphne Looije, and Siamak Hajizadeh

Abstract. In this paper the strategy of Hardheaded negotiating agent is described. Our agent won the Automated Negotiating Agents Competition 2011. As the name implies, the agent is hardheaded, it will not concede until the very end. Useing 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.

1 Introduction

In the competitions, agents perform bilateral multi-issue closed negotiation. That is the agent does not know what the preferences of its opponent are. Hardheaded uses a concession function that over time progressively concedes. Furthermore it is equipped with a learning module that analyzes the bids that are made by the opponent and based on this information assumes that the opponent has a certain preference. Using this knowledge, the agent tries to choose the best bid for its opponent from a set of bids that are equivalent for itself. In this paper the general strategy and particularly the learning function will be described in details. We will conclude with a discussion the agent's performance.

2 Overall Structure

This section describes the structure of our agent. First we describe the general strategy, how the agent decides which bid it will make, and how it gradually concedes

Thijs van Krimpen · Daphne Looije · Siamak Hajizadeh
Delft University of Technology
e-mail: {T.M.vanKrimpen,D.Looije,S.Hajizadeh}@student.tudelft.nl

T. Ito et al. (Eds.): Complex Automated Negotiations, SCI 435, pp. 223–227.
springerlink.com

the minimum utility of accepting bids. Then the learning function is described in more detail. Finally the choice of the parameters is motivated.

2.1 General Strategy

At the beginning of a negotiation session, the agent starts with computing all possible bids and their utility for itself. These are stored in a search tree for rapid retrieval. In a normal domain, 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 from [1] determines the concession limit.

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

$$F(t) = k + (1 - k)\left(\frac{\min(t, T)}{T}\right)^{\frac{1}{e}} \tag{2}$$

Here t is the current time as a proportion of total negotiation time T and k multiplied by the size of the interval [MinUtility − MaxUtility], determines the initial minimum acceptable utility. 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. HardHeaded will give preference to this bid over the ones generated by the agents own learning module, provided that the opponents best bid is above the current reservation value.

2.2 Learning Module

HardHeaded negotiates in domains containing 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 U_b and is the weighted sum of each of the selected values in b.

$$U_b = \sum_{n=1}^{N} w_n \times v_{n,i} \tag{3}$$

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 utillity 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.

Listing 6. update rule

```
if T == 1 then
    Initialize preference profile
else
    set index to where V(T) == V(T − 1)
    W(index) = W(index) + ε
    W = W / Σ_{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. Figure 1 shows a testing of the learning algorithm against an agent which uses an exponential concession function. The agent draws a bid from a range of 300 closest bids above the concession value and offers 1500 bids in total. The utility of all the possible bids which can be generated from the preference profile, is compared to the utility estimated for the same bids by the learning module using $\varepsilon = 0.2$. We have run the above scenario 100 times.

The learning function always manages to identify the most valuable bid, and the least valuable bid for the opponent. However it generally undervalues the opponents bid's utility by roughly 0.2. This offset is not important provided that the order is correct. The average standard deviation of the estimated utility is 0.13, but the mode 0.065. The standard deviation diminishes towards the extremities of the utillity space i.e. in the neighborhood of utility 1 and 0, as is demonstrated in figure1. It also illustrates how the function tends to overvalue certain bid configurations causing a gap or a spread. The overall order of estimated utilities matches the actual opponent's utility function well in this case. There are situations however that this fine auto-adjustment does not occur. 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.

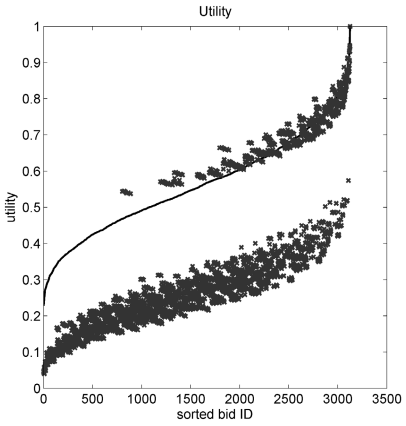

Fig. 1 The solid line shows the utility as generated by the preference profile, the dark x's show the learned utillity.

2.3 Setting Parameters

The agent's performance is dependent on several factors, the domain on which it plays, the opponents, and the computer it runs on. As such the parameter values can only be fine-tuned to a certain extent without losing generality. The parameters which needed tuning were the initial concession k, the discount rate coefficient e, and minimum acceptable utility $MinUtility$. The agent was tuned on domains and against agents from previous ANAC competitions, on three domains selected in one case for it sheer size (Itex vs. Cypress), in another for its zero-sum like preference profiles (inheritance domain), and the third as a generic case (car domain). The agent was tuned against three other agents, agent K, negotiator agent, and IAmHaggler. It was discovered that against agent K a large initial concession increased its willingness to concede later in the negotiating session. Due to this the initial concession k was set at 0.05 or such that at least 5 bids can be made.

Tuning $MinUtility$ and e is a balancing act between holding out as long as possible to get the best negotiation result, and conceding rapidly enough to prevent the agent from timing out. For the ANAC competitions we set $\min_{util}$ value of 0.58, and a concession rate coefficient of 0.02 by only testing our agent against other available agents and in several available domain. This values however are ad hoc and may not result the same achievements in different domains and settings.

2.4 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. The solution presented here was inspired again by Fatima *et al* [1]. As mentioned earlier, The agent creates a virtual deadline at $T_{max} \times \delta$ where δ indicates the discount factor. Also a dynamic minimum utility threshold is calculated according to equation 4.

$$DynamicMinUtility = (MaxUtility - MinUtility) \times \delta + MinUtility \quad (4)$$

3 The Competition

HardHeaded won ANAC2011. Oddly enough, HardHeaded performed better in the final rounds than in the qualifying rounds. There were few notable things in the results of the qualifying rounds. Most important was the number of time-outs, overall about 33%. But when only looking at the results of the qualifying rounds against the other finalists, about 25% of the negotiations ended with a time-out. Furthermore, the average utility reached was higher against only finalists than against all participants in the qualifying rounds. This could explain the better performance in the final rounds.

4 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

1. Fatima, S.S., Wooldridge, M., Jennings, N.R.: Optimal Negotiation Strategies for Agents with Incomplete Information. In: Meyer, J.-J.C., Tambe, M. (eds.) ATAL 2001. LNCS (LNAI), vol. 2333, pp. 377–392. Springer, Heidelberg (2002)