Agent 5: An Agent Using Adaptive Negotiation Strategy and Building User Model Based on Genetic Algorithm

Hanhan Wu (32687001)
Huahua Lin (32930178)
Running Wang (31713831)
Yang Liu (33236437)

Abstract

Agents may need to negotiate together to reach an agreement in some domains. In this paper, we present a bilateral negotiation software agent using ABiNeS as the negotiation strategy and use a genetic algorithm to build a user model. Our agent also participated in agent negotiation competition based on the GENIUS framework.

1 Introduction

In negotiation, each agent has a preference profile. The general purpose of the negotiation on different domains is to maximise the agent's own utility under the premise of reaching an agreement. However, there are uncertainties in negotiation. The opponent's negotiation strategy and preferences are unknown, even the preferences of our user. The background of this report is a bilateral negotiation competition using the Stacked Alternating Offers Protocol (SAOP). We will describe how we build a negotiation agent from different aspects: opponent model, user model, and accepting and bidding strategy. According to the requirements of the competition, the agent's performance is not only concerned with utility but also to minimise the Euclidean distance between the agreement and the Nash bargaining solution. Therefore, a lot of experiments are done to meet these requirements.

2 Opponent Model

2.1 Motivation

According to the work of Tim Baarslag et al. [Baarslag *et al.*, 2013], on one hand, in the case of unpredictable agents, all value models and frequency models perform better and steadily. On the other hand, in the case of predictable agents, value models and frequency models perform much better on accuracy than Bayesian models and theoretical baselines at the beginning but became less accurate over time. One exception is the CUHKAgent value model which performs steadily across all time and its accuracy is always higher than others. CUHKAgent value model is a implementation of ABiNeS [Hao and Leung, 2012]. Hence, we took ABiNeS as our opponent strategy.

2.2 Model Description

2.2.1 Basic Assumptions

The first assumption is that the opponent is rational and will bid according to its concession strategy. Another assumption is that the opponent bids in descending order of preference. Therefore, we can make an inference that a value of an issue, the more frequent and earlier it appears in the negotiation, the more contribution it makes to the opponent's overall utility. However, in the conceding negotiation, the actual utility of the outcome of an agreement before the deadline is discounted by the discounting factor over time, which means

$$U_i^t(\omega) = u_i(\omega)\delta^t. \tag{1}$$

Here, ω is the outcome, δ is discounting factor and t is time.

2.2.2 History of Opponent's Bids

The records of negotiation outcomes proposed by the opponent to the user during a certain time.

2.2.3 Best Bid Prediction for Opponent

Count the number of values of each issue appear in the top 100 unique bids proposed by opponent [Baarslag *et al.*, 2013], and update it every time when there is a new bid from opponent is received, we denote this number as n($\omega(m_i)$) and the accumulated frequency of each outcome as

$$f(\omega) = \sum_{m_i} n(\omega(m_i)). \tag{2}$$

The greater the f(ω) is the more preferred the bid is by the opponent. Meanwhile, we will also adopt ϵ-greedy exploration mechanism to decide whether choose the best bid of a random bid to the opponent randomly, which can make further exploitation of prediction of the opponent.

2.3 Acceptance-Threshold (AT)

Acceptance-threshold is used in accepting, bidding and terminating strategies. It shows the degree of agent's concession and will be updated in each turn. The original formula of updating threshold is as follows,

$$l_A^t = \begin{cases} u^{max} - (u^{max} - u^{max}\delta^{1-\lambda})(\frac{t}{\lambda})^{\alpha} & \text{if } t < \lambda; \\ u^{max}\delta^{1-t} & \text{otherwise.} \end{cases} \tag{3}$$

Here, u^{max} is the maximum utility of user and α is the controller of how the acceptance-threshold approaches $u^{max}\delta^{1-t}$ (boulware($\alpha > 1$), conceder($\alpha < 1$) or linear ($\alpha = 1$)).
The updating formula of λ is as follows:

$$\lambda = \begin{cases} \lambda_0 + (1-\lambda_0)\delta^{\beta}, \text{if } t = 0; \\ \lambda + w(1-\lambda)\sigma_t^{\gamma}, \text{if } 0 < t \leq 1. \end{cases} \quad (4)$$

Here, λ_0 is the minimum value of λ. β determines the way λ varies with the discounting factor δ. σ_t is the ratio of new bids within the recent history of opponent's bids to estimate its degree of concession at time t. γ determines the way λ varies with σ_t. w adjusts the relative effect of σ_t on point λ.
From Fig.1 we can see that no matter how λ it is, the trend of the curve of acceptance-threshold is decreasing when t$< \lambda$, and increasing otherwise.

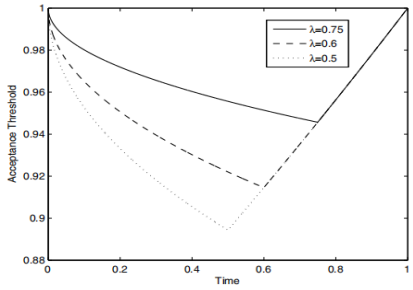

Figure 1: The dynamics of the acceptance-threshold (u^{max} = 1, $\alpha = 0.5$ and $\delta = 0.8$) [Hao and Leung, 2012]. When $\lambda \geq t$, the threshold increases.

According to experiment results, the continuous reduction of the acceptance-threshold is conducive to reaching more agreements, especially for some aggressive agents. Because these agents would rather not make an agreement than reduce their acceptance-threshold. Since both the number of agreements and the final utility of each agreement will be taken into account in the final score, we set the acceptance-threshold to be continuously reduced, but roughly controlled within a range, so as to balance the number of agreements and the utility of each agreement. Therefore, we only took the first half of the acceptance-threshold updating formula instead of the whole, and control the range of the threshold by adjusting variables in the formula.

3 User Model

In most cases, the agents do not have full knowledge about their users. Therefore, a user model should be guessed to obtain the user's utility function. The setting of preference uncertainty allows the agent to have a certain number of ranked bids for the user in advance.

3.1 Motivation

Our idea is to estimate an additive utility space that can represent the users' preferences as accurate as possible based on preference uncertainty. This can be considered as an optimisation problem of finding the best representation with time limitations. Therefore, we utilise the genetic algorithm which has been proved to be a robust and efficient algorithm in solving large-scale searching problems based on the evolutionary theory of "survival of the fittest".

3.2 Genetic Algorithm

For genetic algorithm, the main goal is to search the additive utility space that fits the given ranked bids most by evolution. The main process of the algorithm is shown in Fig.2. At first, the population consists of a certain amount of individuals is initialised. In each generation, the fitness value is served as the evaluation of each individual in the current population. Then, the roulette wheel selection is used to retain a certain number of individuals as parents according to their fitness. After the step of crossover, a certain number of children is added to the next population. This whole process iterates until the end of the set time. Finally, the individual with the best fitness will represent the user's preferences.

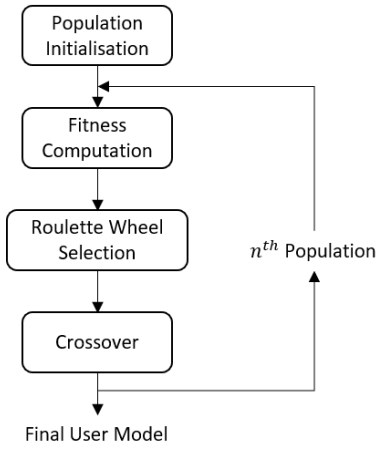

Figure 2: The process of genetic algorithm.

3.2.1 Population Initialisation

The population initialisation module is to generate 1500 individuals in the first generation. The weights for each issue in the specific domain and the evaluation of each value in each issue are randomly initialised, which has the advantage of making the population diverse.

3.2.2 Fitness Computation

Fitness is an index to measure the individual's adaptation to the environment. The higher the fitness value of the individual, the higher the accuracy of the utility space represented by the individual. In our fitness function (as is shown in Eqn.5), we calculate the number of false ranking bids as the $error$ by summing the differences between the ranks of bids outputted by the individual and the real ranks of bids provided by preference uncertainty and then normalised it by dividing the size of bid ranking as our variable. Since the correlation between the variable and the fitness value is negative, we then re-scaled the variable by a Gaussian function with the mean of 0 and the sigma of $0.5 * size$ (to let most of the points falling in the $1 * \sigma$ and $2 * \sigma$ range where the difference in

values is more pronounced).

$$32 \times exp\left\{-\frac{1}{2\sigma^2}(\frac{error}{size}-\mu)^2\right\}. \quad (5)$$

3.2.3 Roulette Wheel Selection

Individuals who are better adapted to their environment (higher fitness) will have a better chance to be selected as parents. Roulette wheel selection satisfies the principle that each individual's probability of being selected is proportional to its fitness as is shown in Eqn.6. Firstly, a random number is generated. The cumulative probability is then calculated for each individual in turn. If the random number is smaller than the current cumulative probability, this individual is selected. This process is repeated 500 times to select one-third of the individuals from the current generation. Besides, 10 additional individuals with the highest fitness in the current population are directly reserved for the next generation.

$$p_i = \frac{f_i}{\sum_{n=1}^{N} f_n}. \quad (6)$$

3.2.4 Arithmetic Crossover

The child is generated by the linear combination of two randomly selected different parents, which is called an arithmetic crossover [Soni and Kumar, 2014]. The weight of each issue for the child is calculated by the sum of weighted issue weights from two parents, where the weights of two parents are normalised fitness values of them. Similarly, the evaluation of each value in each issue is calculated by the sum of weighted evaluation from two parents. Both weights and evaluations have a certain probability of mutation to maintain the diversity of the population. At the early stage of evolution, the mutation rate should be kept high so that the solution space can be fully explored to maintain the diversity of the population and avoid falling into the local optimum. With the increase of generations, the mutation rate should be reduced and stabilised to ensure the convergence of the algorithm because the exploration of solution space has been more sufficient. To realise this idea, we adopt a linear decreasing method to reduce the mutation rate. Specifically, the probability of mutation is set at 0.1 at the very beginning and then decreased by 0.001 with each generation until it is less than or equals 0.04.

3.3 Experiment

3.3.1 Mutation Probability

The basic implementation of a genetic algorithm is to adopt a fixed mutation rate, but the genetic algorithm we implemented adopts the mutation rate which decreases to the middle evolution. In order to compare the difference between the two methods under the premise that other conditions remain unchanged, the metrics of convergence speed and error are utilised. As is shown in Fig.3, the algorithm with decreasing mutation rate converges faster and has less error than the algorithm with a fixed mutation rate.

In order to find the appropriate decreasing rate of mutation, different discounts of decreasing are compared in the condition of same upper (0.1) and lower (0.04) bounds of mutation rate, as shown in Fig.4. The best performance is achieved when the discount of decreasing is 0.001.

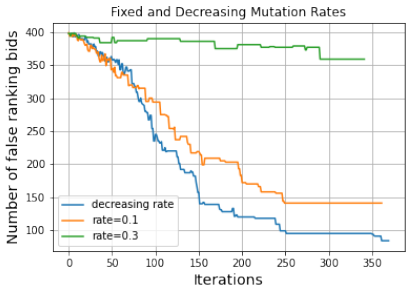

Figure 3: The influence of fixed and decreasing mutation rates.

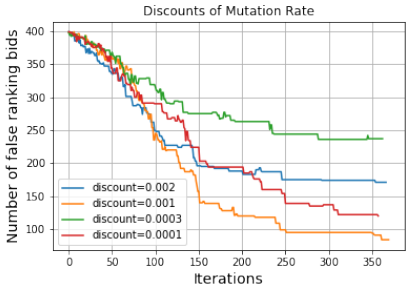

Figure 4: The influence of decreasing rate of mutation.

3.3.2 Algorithm Convergence and Robustness

After finding the appropriate parameters, our genetic algorithm is tested for convergence and robustness in different domains. The individual with the highest fitness value in every generation is selected and their accumulative absolute error between the real rank and the predicted rank is calculated as the evaluation. If the evaluation value shows a decreasing trend among generations in general, it indicates that the individuals in each generation are converging to the optimal additive utility space. Some of the results are shown in Fig.5. We can see that our algorithm performs stably under different domain settings. The curves show that the algorithm converges rapidly in the early stage and can converge for about 350 iterations.

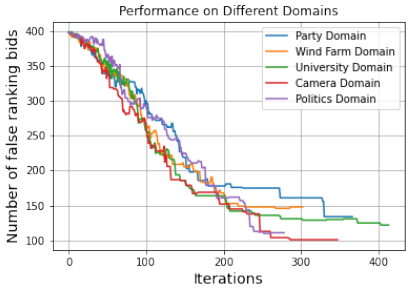

Figure 5: Performance on different domains.

4 Strategies

4.1 Overall Process

a. The first bid proposed by user is the maximum bid in the utility space.
b. Store the bids proposed by opponent and calculate the frequency of values of each issue in the bids after receiving.
c. In each turn, whether the offer can be accepted or terminated will be separately calculated.
d. When the time is far away from the deadline, the offer can be accepted only when it is acceptable and cannot be terminated, or both of them but the utility of this offer is higher than the current reservation utility; The negotiation will be terminated only when the offer is not acceptable and needed to be terminated, or it is acceptable and needed to be terminated but the utility of this offer is less or equal than the current reservation utility.
e. When the deadline is coming, besides the two situations above, if the bid proposed last time is the best bid received from the opponent, the offer can be accepted.

4.2 Bidding

4.2.1 Update Acceptance-Threshold

a. In the case of a large domain or the deadline is coming, the threshold will be given directly by using a value similar to the approximate estimation of the final threshold to reduce the time used by calculation and increase the time left for further negotiation to increase the possibility of reaching an agreement.
b. Otherwise, the threshold will be updated according to section 2.3.

4.2.2 Bid Selection

If the bid with maximum utility in the history of the opponent's bids can not be accepted, then
a. if the maximum utility in the history of opponent's bids is greater than the acceptance-threshold, this bid will be selected. If not, find a bid in the user's utility space with the biggest accumulated frequency according to the opponent model.
b. select 200 random bids between the acceptance-threshold and maximum utility according to step a, and select one of them according to section 2.2.3.
If the bid with maximum utility in the history of the opponent's bids can be accepted, then return it directly.

4.3 Accepting

a. If the utility of the current bid received is greater than the current threshold or the utility of the next bid user will propose, then accept this bid.
b. If the utility of the current best bid user received in the history of opponent's bids is greater than discounted the maximum utility or discounted current acceptance-threshold, and the utility of current bid is approximate to the current best bid, then accept this bid.
c. If neither of the conditions above is satisfied, the bid will be rejected and the bid with maximum utility in the history of the opponent's bids will be proposed in the next turn of negotiation.

4.4 Terminating

a. If the reservation value is greater than acceptance-threshold or the utility of the next bid user will propose, then the negotiation can be terminated.
b. If the discounted reservation value is greater than the discounted maximum utility of the user, then the negotiation can be terminated.

4.5 Experiment

According to the acceptance-threshold updating formula, discounting factor (δ) makes more impact on acceptance-threshold change. To get the optimal discounting factor, we compare the performance of our agent with several opponents in different domains and some results in the party domain are shown in Tab.1. We found that an extremely high discounting factor leads to a bad outcome on Nash point distance and even cannot reach an agreement. The best discounting factor we choose is 0.77 which achieves good results in utility and Nash point distance.

Parameter	AG.#	$\mathbf{U_{opp}}$	$\mathbf{U_{our}}$	Dist. to NP
0.5	5	0.98	0.73	0.08
		0.67	0.94	0.31
		0.49	0.97	0.47
		0.77	0.80	0.16
		1	0.66	0.15
0.77	5	0.74	0.88	0.21
		0.49	0.97	0.47
		0.79	0.82	0.15
		0.79	0.82	0.15
		0.30	1	0.67
0.95	3	0.30	1	0.67
		0.30	1	0.67
		0.30	1	0.67
		0	0	1.22
		0	0	1.22

Table 1: The influence of different discounting factors.

5 Evaluation of Our Agent

We ran our agent in the tournament with several agents in different domains based on the best parameters we have explored in the experiments, and some of the average results are shown in Tab.2. We can observe that not only can our agent achieve higher utility against other agents, but also the agreement is close to the Nash point.

Agent 1	Agent 2	$\mathbf{U_{opp}}$	$\mathbf{U_{our}}$	Dist. to NP
Boulware	Agent	0.71	0.86	0.25
Conceder	Agent	0.51	0.96	0.45
AgentNP1	Agent	0.77	0.83	0.13
Agreeable Agent	Agent	0.74	0.81	0.16
Agent Herb	Agent	0.32	1	0.64

Table 2: Agent evaluation.

6 Conclusions and Future Work

The results of the tournament simulation show that our agent can get higher utility against other agents in most cases while minimising the Euclidean distance between the agreement and the Nash bargaining solution in the meantime on different domains. However, from the results of the competition, there were a few occasions when our agent failed to reach an agreement with our opponents, which indicated the acceptance threshold was a little too high and the pace of concessions was a bit slow for some opponents. Besides, in some cases, our genetic algorithm in the user model is likely to fall into the local optimum in the later stage of evolution. Though it will break the tie after several iterations, it is still an annoying problem that may consume time and computation resources. How to jump out of the local optimum and continue to converge at a faster speed is a question worth considering.

Word Count: 2483

References

[Baarslag *et al.*, 2013] Tim Baarslag, Mark Hendrikx, Koen Hindriks, and Catholijn Jonker. Predicting the performance of opponent models in automated negotiation. In *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 2, pages 59–66. IEEE, 2013.

[Hao and Leung, 2012] Jianye Hao and Ho-Fung Leung. Abines: An adaptive bilateral negotiating strategy over multiple items. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 2, pages 95–102. IEEE, 2012.

[Soni and Kumar, 2014] Nitasha Soni and Tapas Kumar. Study of various crossover operators in genetic algorithms. 2014.