Ava: An Intelligent Agent

Steven Larry Ball*[1], Brent De Hauwere*[1], Matthew De Vries*[1]
*All authors contributed equally to the writing and research in this study. Their names are listed in alphabetical order. [1]

[1]Department of Electronics and Computer Science (ECS), University of Southampton

November 2019

Abstract

We present an agent developed to participate in a bilateral, multi-issue, time-based negotiation tournament using Stacked Alternating Offers Protocol (SAOP) with uncertainty of the user's preference and uncertainty of the opponent's preference. The goal was to maximise the user's utility while minimising the distance to the Nash equilibrium. We propose a Linear Programming approach to dealing with user preference uncertainty and a frequency model for dealing with opponent preference uncertainty. The agent was independently tested against previously achieving agents and the results from the tournament were analysed which present a high performing negotiating agent.

Keywords preference uncertainty, negotiation, SAOP, Linear Programming

1 Introduction

Agent based negotiation is attracting significant attention from the research community in recent years especially given the rise of Artificial Intelligence (AI) that promises to automate most repetitive aspects of our lives [19]. Applications of automatic negotiation using intelligent agents include Wi-Fi channel assignment [1], agriculture supply chain support [10], and e-commerce negotiation [29].

This paper presents a negotiating agent's design and strategy for the University of Southampton's Intelligent Agents module (COMP6203). The negotiation was run using the GENUIS framework [17] and consisted of numerous bilateral negotiations using Stacked Alternating Offers Protocol (SAOP). These negotiations were conducted using various multi-issue preference domains of different sizes. In the negotiation tournament setup used for the coursework there was preference uncertainty. Specifically, the agents had a ranking of a limited set of outcomes, and no access to their entire utility function. However, the agent was able to elicit additional information from a virtual user at a fixed cost. For each domain, there were two different elicitation costs; high and low. The agents also had no knowledge of the preferences of their opponents.

The agent is described in terms of its preference elicitation, opponent model, acceptance strategy and biding strategy. We discuss reasons behind the specific design of the strategy that our agent employed, and how it compares to existing approaches as well as present the results from our testing phase. Finally, we present the results from the tournament with a discussion of possible improvements.

2 Design of the Strategy

When considering the design of the agent's strategy, one needs to take much into consideration such as application and environment in which the agent will run. We present our agent's strategy with reasons and comparisons to existing approaches.

2.1 Preference Elicitation

2.1.1 Theory and Formulation

A major obstacle in the future of representative automated negotiation is the agent's level of knowledge about the preferences of the user it represents [2]. Preference elicitation is a tedious procedure to the users since they have to interact with the system repeatedly and participate in lengthy queries [26]. With elicitation costs, the agent must strike a balance between user model accuracy and user interference. Many strategies have been proposed [3, 4, 2, 18] but most automated negotiation research has focused on opponent preference modeling rather than on the user preference elicitation, however several techniques in opponent modeling are of interest [14].

Baarslag et al. (2019) [26] suggested a linear programming approach which was inspired mainly by the work of Srinivasan and Shocker [25], who proposed a strategy for estimating the weights of different attributes given a set of pairwise comparisons of outcomes by using linear programming to dealing with preference uncertainty. Baarslag et al. (2019) extend this model to propose a different formulation of the problem using categorical data that estimates complete preference profiles based on the outcome set. The process used is summarized.

It is necessary to first define the negotiation domain. During a negotiation, the participants are trying to reach an agreement over m issues which we denote as $I = \{1, \ldots\ldots, m\}$. Every issue, i, is discrete such that each issue can take a finite number of n_i values which we denote as:

$$V_i = \left\{x_1^{(i)}, x_2^{(i)}, \ldots\ldots, x_{n_i}^{(i)}\right\}. \tag{1}$$

The negotiation domain $\Omega = V_1 \times V_2 \times \ldots \times V_m$ is the set of all possible negotiation outcomes. A negotiation outcome $\omega \in \Omega$ is thus an m–tuple that assigns a single value $\omega_i \in V_i$ to every issue i.

2.1.2 A Linear Programming Approach

If an agent is to operate under preference uncertainty, it needs to formulate a strategy that will be able to derive a utility function from a set of pairwise comparisons of outcome. Every user participating in a negotiation has a specific set of preferences regarding the possible outcomes. A preference profile is given by an ordinal ranking over the set of possible outcomes. An outcome ω is said to be weakly preferred over an outcome ω' if $\omega \succeq \omega'$ and strictly preferred if $\omega \succ \omega'$, where $\omega, \omega' \in \Omega$, the negotiation domain. Preference profiles can be expressed in a cardinal way through the use of a utility function such that: $\omega \succeq \omega' \iff u(\omega) \geq u(\omega')$

We focus on linear additive utility functions such that every issue, i's, value is calculated separately according to an evaluation function v_i as follows:

$$u : \Omega \mapsto [0,1] \subseteq \mathbb{R} \quad \text{with} \quad u(\omega) = \sum_{i=1}^{m} w_i \cdot v_i(\omega_i), \tag{2}$$

$$\text{where} \quad \sum_{i=1}^{m} w_i = 1. \tag{3}$$

Here, the w_i represent the normalized weights which indicate the importance of each issue to the user, and $v_i(\omega_i)$ is the evaluation function that maps the i^{th} issue value to a utility.

Consider a ranking of outcomes $\mathcal{O}$, provided in each negotiation, such that

$$\mathcal{O} = \left\{ o^{(1)}, o^{(2)}, \ldots, o^{(d)} \right\}, \tag{4}$$

and the set $\mathcal{D}$ of corresponding pairwise comparisons. From the definition of the utility function, we can integrate the weight and each evaluator value in one variable and we rewrite 2 as:

$$u : \Omega \mapsto [0,1] \subseteq \mathbb{R} \quad \text{with} \quad u(\omega) = \sum_{i=1}^{m} \phi_i(\omega_i), \tag{5}$$

$$\text{with} \quad \phi_i(\omega_i) = w_i \cdot v_i(\omega_i). \tag{6}$$

This gives rise to a new set of discrete variables:

$$Y = \left\{ \phi_1\left(x_1^{(1)}\right), \ldots, \phi_1\left(x_{n_1}^{(1)}\right), \ldots, \phi_m\left(x_1^{(m)}\right) \ldots, \phi_m\left(x_{n_m}^{(m)}\right) \right\}. \tag{7}$$

One more piece of information is needed to formulate the problem of estimating the utility function as a linear optimization problem with the set Y as the unknown variables. For each pairwise comparison between outcomes $(o, o') \in \mathcal{D}$, we derive that:

$$\sum_{i=1}^{m} \left(\phi_i(o_i) - \phi_i(o_i')\right) \geq 0, \quad \text{with} \quad \phi_i(o_i), \phi_i(o_i') \in Y. \tag{8}$$

We the make the definition:

$$\Delta u_{o,o'} = \sum_{i=1}^{m} \left(\phi_i(o_i) - \phi_i(o_i')\right), \quad \Delta u_{o,o'} \geq 0. \tag{9}$$

Finally, we can translate the above inequalities into a linear optimization problem. We need to define 'slack' variables, z, such that the number of 'slack' variables, $z_{o,o'}$, is equal to the number of comparisons (o, o') in $\mathcal{D}$.

Now, we are able to formulate the linear program as:

$$\text{Minimize:} \quad F = \sum_{(o,o') \in \mathcal{D}} z_{o,o'}, \tag{10}$$

Subject to the following:

$$z_{o,o'} + \Delta u_{o,o'} \geq 0, \tag{11}$$

$$z_{o,o'} \geq 0, \text{ for } (o, o') \in \mathcal{D}, \tag{12}$$

$$\phi_i\left(x_j^{(i)}\right) \geq 0, \text{ for } i \in I, j \in \{1, 2, \ldots, n_i\}. \tag{13}$$

The decision variables are $Y \cup \{z_{o,o'} \mid (o, o') \in \mathcal{D}\}$. In order to avoid the trivial solution to the problem in its current form, an additional constraint is needed. In our implementation and that in [26], the additional information is the outcome of maximum utility for the user, ω^*. This gives rise to one final constraint for the problem:

$$u(\omega^*) = 1 \quad \Rightarrow \quad \sum_{i=1}^{m} \phi_i'(\omega_i^*) = 1. \tag{14}$$

2.2 Opponent Model

Although achieving a *Pareto efficient* outcome may be desirable, a negotiation agent has no knowledge of the opponent's preferences, nor does it have knowledge of the negotiation strategy. To address this, a wide range of papers [31, 13, 30, 6, 7, 5, 12] propose algorithms which try to infer the preferences and negotiation strategies of the opponent. In terms of additive utility preferences, this means inferring the weights as well as the utility for individual values for each issue. Approaches differ in terms of complexity ranging from simple heuristic to machine learning approaches. Our agent has made use of a similar technique used by the *Hardheaded* agent [28] which uses a frequency modelling approach.

2.2.1 Hardheadedness

Frequency approaches generally model the opponent's preferences by adding the frequency of issue values and the frequency of updates in the issues of the offered bid, without looking at a specific set of hypotheses [27]. We took this approach and used it to determine how hardheaded an opponent agent is. This is used to avoid not reaching an agreement with agents that concede very slowly which results to lower agreement rates. The variable h represents the hardheadedness and its value will range from 0 to 1.

The hardheadedness is calculated by:

$$h = 1 - \frac{freqIssueValueUpdate}{totalIssues * totalTurns}. \tag{15}$$

2.2.2 Weight of Issues

To calculate the weight of issues, the approach used by the *Jonny Black* [31] agent was implemented. Using this approach it is assumed that the probability is relatively small for an opponent to change from its preferred option for issues with greater importance. We used the *Gini Index* [20] as the impurity measure similar to the Jonny Black agent. Issues with bigger *Gini-Impurity* scores are weighted more by the opponent.

The weight of issues is calculated by:

$$\hat{w} = \sum \frac{frequencyOfIssue^2}{totalTurns^2}. \tag{16}$$

With the calculated $\hat{w}$, the weight is normalised by dividing the unnormalised weight by the total unnormalised weight:

$$w = \frac{unnormalisedWeight}{totalUnnormalisedWeight}. \tag{17}$$

2.2.3 Evaluation of Issues

We took a simple heuristic approach to find the evaluation of issues by using the frequency analysis approach. This was done by setting the issue's evaluation equal to the issue value divided by the issue value with the maximum frequency in the history bid. The variable e is used to represent the evaluation of an issue.

The evaluation of an issue is calculated by:

$$e = \frac{issueValue}{issueValueMaxFreq}. \tag{18}$$

2.3 Acceptance Strategy

This agent combines three acceptance conditions to devise an acceptance strategy:

1. $\mathrm{AC_{const}}(\alpha)$: the agent accepts the opponent's bid if it is higher than the target utility.
2. $\mathrm{AC_{next}}(\alpha)$: the agent accepts the opponent's bid if the utility is higher than the utility of its own proposed bid.
3. $\mathrm{AC_{time}}(\alpha)$: the agent accepts the opponent's bid after a predetermined amount of time.

Cao and Dai (2014) [22] clarify the shape of the concession curve for classic Boulware and Conceder tactics. They further propose a simplified Boulware utility acceptance strategy:

$$AC_{\mathrm{const}}(\alpha) = \frac{\log\left(t_{\mathrm{left}}\right)}{c(t)} + Ka, \tag{19}$$

where $t_{\mathrm{left}} = \frac{\min(t, T\max)}{\mathrm{Tmax}}$.

The numerator expresses the normalised amount of time left. The denominator $c(t)$ acts as a conceding factor that governs the concession rate, and Ka determines the minimum starting utility; both constants will be discussed in the next section.

Equation 20 shows a variant of $\mathrm{AC_{time}}(\alpha)$ we implemented. It factors in a hardheadedness value of the opponent h and the amount of time that is left in the negotiation. Once 90% of the negotiation time has passed, the value of $c(t)$ is decreased to slightly concede. If the opponent is rather hardheaded, the value of $c(t)$ is further reduced. By doing so, we increase the likelihood of reaching an agreement.

$$c(t) = \begin{cases} 13 & \text{if } 0.1 < t_{\mathrm{left}} \leq 1 \\ 10 & \text{if } 0 < t_{\mathrm{left}} \leq 0.1 \land h \leq 0.6 \\ 7 & \text{if } 0 < t_{\mathrm{left}} \leq 0.1 \land h > 0.6 \end{cases} \tag{20}$$

2.4 Bidding Strategy

It is common for agents to offer their maximum utility at the start of negotiations while it builds an opponent model and searches the outcome space [8]. Furthermore, doing this allows for finding bids near the Nash equilibrium. The agent implements this strategy by offering a bid that maximises its own utility for the first 20% of the time.

Every 10 rounds, the agent recalculates the Nash product with the best saved bids from the opponent. This increases the accuracy of the opponent model as it takes into account new offers.

Each round, the agent generates 100 bids above the target utility from $\mathrm{AC_{const}}(\alpha)$. The bid with the best Nash product is kept in a *bestGeneratedBids* array. As soon as the array is full, the worst bid in the list is replaced by the best generated bid if its Nash product is higher.

Because the opponent model is a mere approximation and is used to calculate the Nash product, a top five bid from the array is randomly selected and offered.

3 Testing

After creating the design and strategy, we tested our agent against ten other agents. Most of the agents we negotiated with competed in the *ANAC2015* competition such as *CUHKAgent2015* [23], *Atlast3* [21], *AgentH* [11], *AgentX* [9], *JonnyBlack* [31], *ParsAgent* [15], *PhoenixParty* [16] and *PokerFace* [24].

The negotiation setup was configured the same way as the competition. It was time based, with each negotiation lasting 90 seconds but the domain used for testing was the *party_domain*. Our agent received the highest utility when negotiating with the *ConcederNegotionParty* and *ParsAgent* with a utility of 0.90 from both agents and the lowest with *CUHKAgent2015* and *PhoenixParty* with a utility of 0.32 from both agents. Even though the performance wasn't that good when negotiating with *CUHKAgent2015* and *Phoenix-Party* it was observed that our agent performed well in most negotiations. In Table 1, it can be seen that our negotiations achieved an average utility of 0.73, average distance to Pareto of 0.06 and average distance to Nash equilibrium of 0.24. The table containing the full test results can be found in Appendix A.

Avg Util	Avg Dist to Pareto	Avg Dist to Nash
0.72631	0.05554	0.24012

Table 1: Average Distance and Utility

4 Results

To evaluate the performance of this agent in the competition, we will analyse different measurements. The agent competed in four domains of different sizes: SportHal (SH) with 243 offers, Party with 3072 offers, WindFarm (WF) with 7200 offers, and Energy with 15625 offers. For each domain as well as overall, we will consider the number of agreements, the obtained utility, and the distance to Nash equilibrium.

Overall, the agent performed very well and scored among the best. We obtained an agreement rate of 99.4%, an average user utility of 0.86, and a distance to Nash equilibrium of 0.17. Unfortunately, we discovered that a very small amount of null pointer exceptions occurred during the tournaments. Resolving this issue would result in an even closer to optimal agreement rate.

In Table 2, the agreement rate across the various domains is displayed. We can observe that the agreement rate is generally very high. However, as the size of the domain increases, the agreement rate slightly decreases. In fact, in the largest domain, we obtained a rate of 98.45%. Nevertheless, this percentage reflects the fact that the agent does an excellent job at finding an offer that provides a good utility for its opponent and itself.

	SH	Party	WF	Energy	Total
Agreements	1840	1840	1803	1649	7132
Total	1840	1840	1820	1675	7175
Percentage	100%	100%	99.07%	98.45%	99.40%

Table 2: Agreement Rate per Domain

Figure 1 and 2 respectively show box plots of the obtained utility and distance to Nash equilibrium for each domain. We immediately notice that the highest performance is obtained in the smallest domains. This is not a surprise; the bigger the domain, the more effort that is required to estimate the user and opponent model. However, the agent was able to withstand the more larger domains too.

In the smallest domain, ShortHal, we obtained a median user utility of 0.98 and a median distance to Nash equilibrium of 0.06. This high accuracy is obtainable because it is easy to explore the outcome space and identify bids near the Nash equilibrium. In medium-sized domains Party and WindFarm, the agent achieved a user utility of 0.86 and 0.85, and a distance to Nash equilibrium of 0.1 and 0.11 respectively. In the Energy domain, we obtained a user utility of 0.82 and distance to Nash equilibrium of 0.31. Especially the distance to Nash is notably off in this domain. Aside from the increased difficulty to explore the outcome space, this is partly caused by the hardheaded nature of the agent.

In general, the agent performed exceptionally well and ranked within the top five in the competition. Even though the performance reduced as the domain size increased, the agent found an excellent generic approach to cope with differently sized domains.

5 Discussion

Although the agent performed well in the tournament, improvements could still be implemented.

From the tournament logs, we concluded that our agent runs into a null pointer exception in very rare occasions, namely 40 out of 7175. The first essential improvement would be to find what causes this and resolve it. By doing this, we would steadily improve our agreement rate.

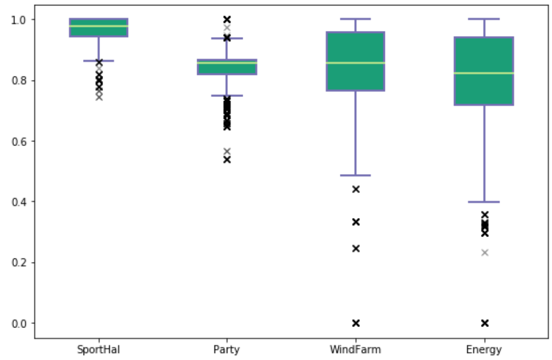

Figure 1: Box Plots of User Utility per Domain

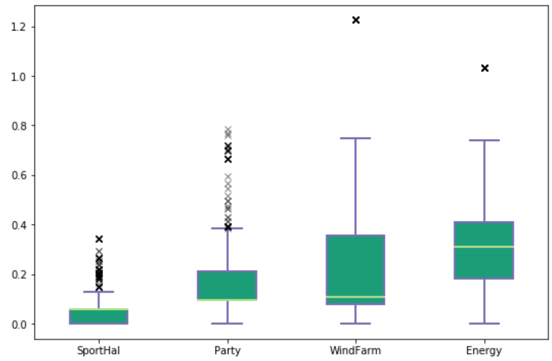

Figure 2: Box Plots of Distance to Nash Equilibrium per Domain

Furthermore, even though the agent did not run into timeouts, our offering strategy is very computationally expensive as it generates 100 bids every round. To increase efficiency and prevent the agent from having timeouts in larger domains, an alternative for the random bid generation should be sought. We thought about this during the development of our agent, but could not find an ideal solution.

A suggestion closely related to the subject above is the *bestGeneratedBids* array. Over time, we could reduce the size of this array instead of keeping it fixed at 100. This would reduce the time needed to search the outcome space and lower the effort to sort the list on utility value. However, this could potentially impact our accuracy and ability to offer the right bids, and given that our agent did not have timeouts, we made the right decision not to do so—in this tournament setup.

6 Conclusions

It is challenging enough to design an agent to negotiate on behalf of a user, but when dealing with preference uncertainty of both the user and the opponent, the problem becomes highly nontrivial. This agent made careful use of Linear Programming and a frequency model to deal with this uncertainty and proved to, together with the acceptance and bidding strategy, maximise user utility and minimise the distance to Nash equilibrium. Matthew De Vries focused on the preference elicitation while Steven Ball focused on the opponent modeling and Brent De Hauwere on the acceptance and bidding strategy.

References

[1] Tim Baarslag, Alper T. Alan, Richard Gomer, Muddasser Alam, Charith Perera, Enrico H. Gerding, and m.c. schraefel. An automated negotiation agent for permission management. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, pages 380–390, Richland, SC, 2017. International Foundation for Autonomous Agents and Multiagent Systems.

[2] Tim Baarslag and Enrico H. Gerding. Optimal incremental preference elicitation during negotiation. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 3–9. AAAI Press, 2015.

[3] Tim Baarslag, Mark J. Hendrikx, Koen V. Hindriks, and Catholijn M. Jonker. Learning about the opponent in automated bilateral negotiation: A comprehensive survey of opponent modeling techniques. *Autonomous Agents and Multi-Agent Systems*, 30(5):849–898, September 2016.

[4] Tim Baarslag and Michael Kaisers. The value of information in automated negotiation: A decision model for eliciting user preferences. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, pages 391–400, Richland, SC, 2017. International Foundation for Autonomous Agents and Multiagent Systems.

[5] Siqi Chen, Haitham Ammar, Karl Tuyls, and Gerhard Weiss. *Optimizing complex automated negotiation using sparse pseudo-input Gaussian processes*, volume 1. 05 2013.

[6] Siqi Chen and Gerhard Weiss. *An Efficient and Adaptive Approach to Negotiation in Complex Environments*, page 228–233. ECAI'12. IOS Press, NLD, 2012.

[7] Siqi Chen and Gerhard Weiss. *A Novel Strategy for Efficient Negotiation in Complex Environments*, pages 68–82. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[8] A. S. Y. Dirkzwager, M. J. C. Hendrikx, and J. R. De Ruiter. *TheNegotiator: A Dynamic Strategy for Bilateral Negotiations with Time-Based Discounts*, pages 217–221. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[9] Wen Gu and Takayuki Ito. *Agent X*, pages 239–242. Springer International Publishing, Cham, 2017.

[10] W. Guo, W. Li, Y. Zhong, G. Lodewijks, and W. Shen. Agent-based negotiation framework for agricultural supply chain supported by third party logistics. In *2016 IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 584–589, May 2016.

[11] Masayuki Hayashi and Takayuki Ito. *AgentH*, volume 674, pages 251–255. 04 2017.

[12] He He, Jordan L. Boyd-Graber, Kevin Kwok, and Hal Daumé III. *Opponent Modeling in Deep Reinforcement Learning*, volume abs/1609.05559. 2016.

[13] Koen Hindriks and Dmytro Tykhonov. *Opponent Modelling in Automated Multi-Issue Negotiation Using Bayesian Learning*, page 331–338. AAMAS '08. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2008.

[14] Catholijn M. Jonker, Valentin Robu, and Jan Treur. An agent architecture for multi-attribute negotiation using incomplete preference information. *Autonomous Agents and Multi-Agent Systems*, 15(2):221–252, October 2007.

[15] Zahra Khosravimehr and Faria Nassiri-Mofakham. *Pars Agent: Hybrid Time-Dependent, Random and Frequency-Based Bidding and Acceptance Strategies in Multilateral Negotiations*, pages 175–183. Springer International Publishing, Cham, 2017.

[16] Max Lam and Ho-fung Leung. *Phoenix: A Threshold Function Based Negotiation Strategy Using Gaussian Process Regression and Distance-Based Pareto Frontier Approximation*, volume 674, pages 201–212. 04 2017.

[17] Raz Lin, Sarit Kraus, Tim Baarslag, Dmytro Tykhonov, Koen Hindriks, and Catholijn M. Jonker. Genius: An integrated environment for supporting the design of generic automated negotiators. *Computational Intelligence*, 30(1):48–70, 2014.

[18] Yasser Mohammad and Shinji Nakadai. Fastvoi: Efficient utility elicitation during negotiations. In Tim Miller, Nir Oren, Yuko Sakurai, Itsuki Noda, Bastin Tony Roy Savarimuthu, and Tran Cao Son, editors, *PRIMA 2018: Principles and Practice of Multi-Agent Systems*, pages 560–567, Cham, 2018. Springer International Publishing.

[19] Yasser Mohammad and Shinji Nakadai. Optimal value of information based elicitation during negotiation. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, pages 242–250, Richland, SC, 2019. International Foundation for Autonomous Agents and Multiagent Systems.

[20] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.

[21] Akiyuki Mori and Takayuki Ito. *Atlas3: A Negotiating Agent Based on Expecting Lower Limit of Concession Function*. 2017.

[22] Mukun Cao and Xiaopei Dai. Multi-strategy Selection Model for Automated Negotiation. In *2014 47th Hawaii International Conference on System Sciences*, pages 250–259, Waikoloa, HI, January 2014. IEEE.

[23] Chi Wing Ng, Hoi Tang Leung, and Ho-fung Leung. *CUHKAgent2015: An Adaptive Negotiation Strategy in Multilateral Scenario*, pages 243–250. Springer International Publishing, Cham, 2017.

[24] J. B. Peperkamp and V. J. Smit. *Pokerface: The Pokerface Strategy for Multiparty Negotiation*, pages 213–218. Springer International Publishing, Cham, 2017.

[25] V. Srinivasan and Allan D. Shocker. Linear programming techniques for multidimensional analysis of preferences. *Psychometrika*, 38(3):337–369, Sep 1973.

[26] Dimitrios Tsimpoukis, Tim Baarslag, Michael Kaisers, and Nikolaos Paterakis. *Automated Negotiations Under User Preference Uncertainty: A Linear Programming Approach*, pages 115–129. 04 2019.

[27] Okan Tunalı, Reyhan Aydoğan, and Victor Sanchez-Anguix. *Rethinking Frequency Opponent Modeling in Automated Negotiation*, pages 263–279. Springer International Publishing, Cham, 2017.

[28] Thijs van Krimpen, Daphne Looije, and Siamak Hajizadeh. *HardHeaded*, pages 223–227. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[29] Gong Wang, T.N. Wong, and Chunxia Yu. Computational method for agent-based e-commerce negotiations with adaptive negotiation behaviors. *Procedia Computer Science*, 4:1834 – 1843, 2011. Proceedings of the International Conference on Computational Science, ICCS 2011.

[30] Colin R. Williams, Valentin Robu, Enrico H. Gerding, and Nicholas R. Jennings. *IAMhaggler: A Negotiation Agent for Complex Environments*, pages 151–158. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[31] Osman Yucel, Jon Hoffman, and Sandip Sen. *Jonny Black: A Mediating Approach to Multilateral Negotiations*, pages 231–238. Springer International Publishing, Cham, 2017.

Appendix A

Agent	**Distance to Pareto**	**Distance to Nash**	**Agent 1 Utility**	**Opponent's Utility**
BoulwareNegotiation	0.02069	0.10210	0.83564	0.84008
ConcederNegotiation	0.00000	0.02446	0.90415	0.78511
CUHKAgent2015	0.11743	0.61448	0.32010	0.87024
Atlas3	0.13180	0.41521	0.51782	0.84521
AgentH	0.14737	0.18412	0.85954	0.61410
AgentX	0.02069	0.04271	0.88767	0.77259
JonnyBlack	0.00000	0.14492	1.00000	0.65867
ParsAgent	0.00000	0.02446	0.90415	0.78511
PhoenixParty	0.11743	0.61448	0.32010	0.87024
PokerFace	0.00000	0.23430	0.71389	0.87856
Average	0.05554	0.24012	0.72631	0.78353

Table 3: The negotiation test results with different agents.