Agent 17

Martin Ingram
Email: mi2n23@soton.ac.uk
Student ID: 35475277

Daniel Turato
Email: dt3n23@soton.ac.uk
Word Count: 2462

Oliver Simmonds
Email: os13g23@soton.ac.uk
Student ID: 35321997

***Abstract*—Agent 17 combines novel ideas with well-established methods in the agent-based negotiation field. Here we introduce an approach in its strategy, which is to hot-swap into specific algorithms at optimal moments. This strategy aims to maximise the agent's utility, whilst also minimising the distance from the Nash equilibrium. Furthermore, we explore techniques for tuning an agent's hyperparameters through various optimisation algorithms.**

I. Introduction

Agent 17 adopts a number of novel techniques and here we attempt to highlight the benefits and drawbacks of these methods with a rigorous evaluation.

Agent 17 has three main functional components:

- The Agent Strategy,
- Opponent Modelling,
- Preference Elicitation.

We discuss each of these below, providing insight into the decision-making, and then focus on how we optimised the agent using hyperparameter tuning.

The agent had some success in the final tournament, but nevertheless to conclude our project we also identify areas for improvements and future work.

II. Strategy

A. Choose Action

The strategy is responsible for handling incoming bids and choosing an appropriate counteroffer. This uses the following variables:

- $t =$ the current number of negotiation rounds
- $o =$ Estimated utility of the opponent's most recent bid
- $r =$ Our reservation value
- $b =$ The best utility from all previous opponent bids
- $m =$ Our concession strategy's minimum utility threshold.

For each negotiation round, the agent assesses the negotiation situation based on the above and selects one of the following seven scenarios.

Scenario 1: If estimated opponent utility o exceeds the threshold m and we are close to the end of the negotiation, we accept. What we consider "close" is determined by a tuning hyperparameter ($finishTime$).

Scenario 2 If estimated opponent utility o surpasses our reservation value r, but falls below our threshold m near the end of negotiations, we may decide to "give up" and simply accept this poor offer. This happens after the time determined by the tuning hyperparameter "$giveUpTime$". If however the best offer we've received (based on b) is 0.1 higher than o, we offer back the bid associated with that best offer instead of giving up.

The next three scenarios generate a potential counterbid using the bidding strategy described in the next section. All the remaining scenarios occur with sufficient time left in negotiations.

Scenario 3 If we receive an offer from our opponent but the potential counterbid would produce higher utilities for both parties, then we offer this counterbid.

Scenario 4 If the estimated utility of the opponent's bid o is below our threshold m, then we offer a counterbid.

Scenario 5 If none of the above scenarios are invoked and the estimated utility of the opponent's bid o is above our threshold m, then this is a great offer and we accept it.

Scenario 6 In this trivial scenario, we generate the first bid of the negotiation.

Scenario 7 If we run out of options then we end negotiations with nothing.

B. Bidding Strategy

Our bidding strategy tries to generate the optimal bid for the opponent above our threshold m. A list of potential bids is generated at the start of the negotiation. If preference uncertainty is disabled then this will be a list of up to 1000 random bids. If it is enabled then we generate a list of bids which we elicit.

If the list of potential bids is empty, we offer a bid which has the highest utility for us. Otherwise we ensure that the size of the list is below a tuning hyperparameter ($maxListSize$) by truncating a shuffled copy of the list. Finally, we find the bid which has the highest opponent utility given our opponent model that is above our threshold m.

C. Minimum Threshold

The minimum threshold m is determined by our concession strategy and employs two popular methods. Initially, we have little information about the opponent's utility space and hence use the common uninformed Boulware concession strategy [1] in which our minimum threshold is determined using a tuning hyperparameter ($boulwareBeta$).

However, during the negotiation we gain an understanding of our opponent's utility space and hence transition to a Tit-for-Tat based strategy [2]. The time at which we transition

is determined by another tuning hyperparameter (transitionTime).

Our adaptation of the Tit-for-Tat strategy estimates how much our opponent has conceded, (if negative, we set this value to zero), and applies a weighting to this value dependent on how far from the estimated Nash Equilibrium (NE) the last offer was. In the aim of reaching an outcome near the NE we concede slower if we believe we are far from the NE. If close we concede at the same pace as our opponent.

III. Opponent Modelling

The opponent modelling algorithms used by Agent17 builds upon those used by the "Jonny Black" agent [3] and also uses the same assumptions:

- The opponent's preference for a particular issue option is reflected in the number of times we receive an offer with that option.
- The relative importance of an issue for the opponent is reflected in their unwillingness to change that selected option.

A. Evaluating Options

We decided to implement a simple linear relationship between the option count and its value to the opponent. This avoided a problem that we observed, whereby the value of options that have never been selected by the opponent were overestimated, particularly in the early rounds of the negotiation.

Hence the value V_o of an option is simply

$$V_o = \frac{f_o}{r},$$

where f_o is the frequency of option o and r the current number of rounds.

B. Weights of Issues

We chose to stick with the same algorithm as Jonny Black [3] for the weight of issues, namely the unnormalised weight $\hat{w_i}$ being

$$\hat{w_i} = \sum_{o \in O_i} \frac{f_o^2}{r^2},$$

where f_o is the frequency of option o and r is the current number of negotiation rounds.

Similarly the normalised weight of each issue w_i is

$$w_i = \frac{\hat{w_i}}{\sum_{j \in I} \hat{w_j}},$$

for the set of issues I.

C. A Worked Example

Given the following frequencies after 25 rounds

	Option 1	Option 2	Option 3	Option 4
Issue 1	17	8	0	0
Issue 2	5	6	7	7

the value of each option will be

	Option 1	Option 2	Option 3	Option 4
Issue 1	0.68	0.32	0.0	0.0
Issue 2	0.20	0.24	0.28	0.28

The weights of each issue using the formula above will be;

$$\hat{w_1} = \frac{17^2}{25^2} + \frac{8^2}{25^2} + \frac{0^2}{25^2} + \frac{0^2}{25^2} = \frac{353}{625} = 0.5648,$$
$$\hat{w_2} = \frac{5^2}{25^2} + \frac{6^2}{25^2} + \frac{7^2}{25^2} + \frac{7^2}{25^2} = \frac{159}{625} = 0.2544,$$

and the normalised weights;

$$w_1 = \frac{0.5628}{0.5648 + 0.2544} = \frac{0.5648}{0.8192} \approx 0.6895,$$
$$w_2 = \frac{0.2544}{0.5648 + 0.2544} = \frac{0.2544}{0.8192} \approx 0.3105.$$

Note that the strong preference for Issue 1, Option 1 is reflected in these values.

Finally for an offer with Option 1 for Issue 1 and Option 3 for Issue 2 then the estimated utility of,

$$V(O_1, O_3) = (0.6895 \times 0.68) + (0.3105 \times 0.28) \approx 0.5558.$$

D. The Recent Bid Window

We also introduced a "bid window", whereby only data from the last n rounds were considered when estimating opponent utility. This is based on the idea discussed in Baarslag et al [4], namely that the opponent may be less likely to vary the selection of important options as the negotiation progresses.

The number of rounds to consider is determined using a hyperparameter $recentBidWindow$. For the final submitted agent, this was set to 10.

IV. User Modelling

Our agent will be competing in a tournament where preference elicitation is enabled. This means that our agent won't have full knowledge of our preferences; we will need to develop an algorithm that infers our preferences at a high degree of accuracy. Furthermore, we will need a strategy that determines if we need to elicit new bids, and at what frequency.

A. Preference Elicitation

We adopted an elicitation strategy first introduced by the $AhBuNeAgent$ [5], the winner of the ANAC 2020 competition. Consider;

$$n_e = \frac{\lambda}{e_c}, \tag{1}$$

$$n_b = \begin{cases} \max\left(\min(0.1 \cdot |\Omega| - |b_a|, n_e), 0\right) & \text{if } |\Omega| \leq 100 \\ \max\left(\min(10 - |b_a|, n_e), 0\right) & \text{otherwise.} \end{cases} \tag{2}$$

The strategy elicits n_b random bids before a negotiation begins, see Eq. 2. Depending on the size of the domain $|\Omega|$, we want the total number of bids in our bid ranking $|b_a|$ to be at least 10% of $|\Omega|$ for small domains, or at least 10 for larger domains.

Additionally, we set a maximum number of bids that we wish to elicit n_e, see Eq 1. This is determined by the elicitation cost e_c and a penalty hyperparameter λ.

Var.	$c_{1,1}$	$s_{1,1}$	$c_{1,2}$	$s_{1,2}$	$\dots$	$c_{1,M}$	$s_{1,M}$
i_1	$c_{2,1}$	$s_{2,1}$	$c_{2,2}$	$s_{2,2}$	$\dots$	$c_{2,M}$	$s_{2,M}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
i_N	$c_{N,1}$	$s_{N,2}$	$c_{N,2}$	$s_{N,2}$	$\dots$	$c_{N,M}$	$s_{K,M}$

Fig. 1: Simplex Method tableau for i_N issues, s_K slack variables, c_M constraints

B. Estimating User Preferences

Estimating our agent's utility space can be modelled as a constrained minimisation problem [6], which can be solved using linear programming methods. We decided on solving this problem through the Simplex method [7].

Using our agent's partial preference order, we first build a tableau that determines each utility for i_N issues in our domain, see Fig. 1. Alongside our issues, we provide s_K slack variables for each bid in our bid ranking. With our decision variables chosen, we then set multiple constraints. The constraints restrict the values of each issue to satisfy the preference order and ensure they're between the maximum and minimum possible bids in the domain.

Similarly, we then produce a secondary tableau using the calculated utilities, to calculate w_N weights for each issue. Using both the utilities and weights, we can initialise our agent's utility space before a negotiation begins.

V. Testing

Our goal was to test our agent against challenging opponents, to understand its weaknesses. So, we opted to run tournaments against three podium winners from the ANAC 2018 competition, Agent33, AgreeableAgent2018, and Agent36 in a variety of domains.

Though it was trivial to run single tournaments, we wished to run consecutive tournaments whereby our agent adjusted the hyperparameters highlighted in sections II - IV based on previous performances. To accomplish this, we developed an automated script that would run a hyperparameter optimisation algorithm over N_t given tournaments.

A. Random Search

We initially performed a random search [8] on the hyperparameter solution space, to locate a selection of parameters that maximised our utility gained.

Let $P = \{i_1, i_2, \dots, i_n\}$ be the set of parameters in our agent, P' be the set obtained by sampling each variable i from a uniform distribution between its lower bound i_l and upper bound i_u, The parameters are given by:

$$P' = \{P'_i \mid P'_i \sim \text{Uniform}(i_l, i_u) \text{ for each } i \in P\}.$$

Then, for each tournament t_i up to N_t, we update P' and record our agent's total utility gained from each negotiation it participated in.

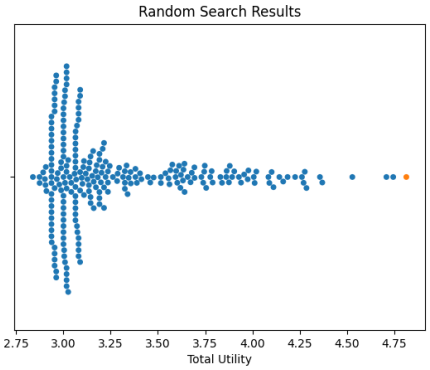

Fig. 2: The results of the random search, visualised using a Swarmplot. Each data-point represents the total utility gained in a single tournament.

Due to the randomness and high-dimensional solution space, it was difficult to find a combination of parameters that maximised our utility, see Fig. 2. However, we did encounter a few outliers, which seemed to perform better than the majority of outcomes.

B. Bayesian Optimisation

Bayesian hyperparameter Tuning attempts to build a model of an objective function, which in our case was the average utility received in the test tournament [9]. It does this by employing Bayes formula:

$$Pr(T|P) = \frac{Pr(P|T) \cdot Pr(T)}{Pr(P)},$$

where T represents the objective function and P the set of hyperparameters.

This informed approach is much more efficient when the hyperparameter space is large and hence we employed it to determine the hyperparameters we took into the class tournament.

VI. Evaluation

A. Tournament Performance

To start to understand the abilities of our model we compare it to other agents in the class tournament results. Our first observation being that Agent17 handled all domains robustly, regardless of the size of the domain, the number of known offers and the elicitation cost. We only failed negotiations when opposing Agent3, in the two larger domains and with a low number of known offers. In both cases Agent3 halted the negotiation.

We note that there were six agents overall that had technical issues of some sort. We have allowed for this in our subsequent analysis of the data in the tournament logs to ensure it does not skew the statistics generated. See Fig. 3 for our understanding of the tournament results.

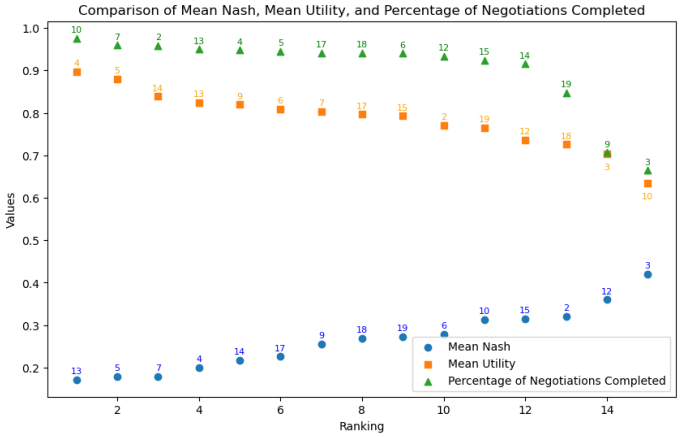

Fig. 3: The class tournament results with respect to; distance from NE, mean utility, and percentage of negotiation agreements. The label on each data point represents the agent number. Agents in failed negotiations were given a distance from NE score of 1.

As shown in Fig. 3, Agent17 had the 7^{th} highest proportion of negotiation agreements, slightly above average. Only three agents showed inconsistency in reaching agreements and hence Agent17 is typical on this metric.

Agent17 ranked 6^{th} and 8^{th} for mean distance from the NE and average utility acquired respectively. With 8^{th} being the average position, both these metrics show that our agent was strong but that there is room for improvement. Interestingly, we performed better with respect to distance from the NE despite tuning our agent for optimal utility.

In future work, we would investigate how tuning on the NE instead may have impacted this result. Our success in ending near the NE may be product of our concession strategy wherein we conceded less when the opponent offered bids far from the NE. Our agent's ability to effectively predict the location of the NE is credit to our opponent modelling and preference elicitation.

In Fig. 4 and 5 we investigated the results in more depth by splitting the analysis up by domain size.

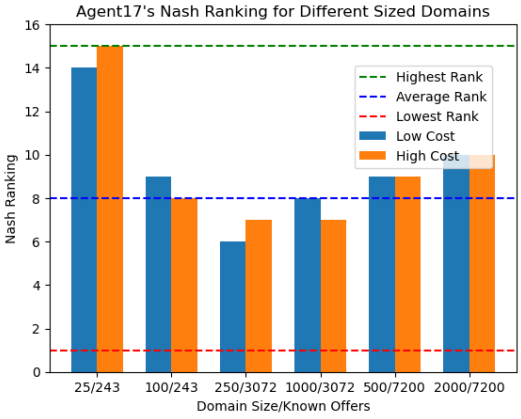

Fig. 4: A plot showing Agent17 performance in different domain sizes with respect to distance from the NE. Note, in this plot, the higher the ranking the better.

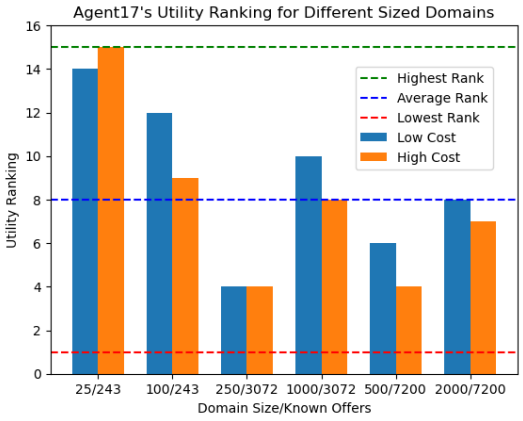

Fig. 5: A plot showing Agent17 performance in different domain sizes with respect to utility acquired. Note, in this plot, the higher the ranking the better

Both of the above plots demonstrate that our strongest performance was in small domains. We hypothesise that this was due to our approach to hyperparameter tune our models on the genius domains available to us, which were all small domains.

These results encourage us to believe that hyperparameter tuning was a good approach but was not applied to its full ability in this scenario. We acknowledge that there is a possibility that our model overfitted to small domains.

If we had access to larger domains and more time we would have tuned on them as well and likely seen an improved performance. Future work could involve building a model capable of distinguishing between different domain sizes and adapting the hyperparameters used accordingly. This way we could tune hyperparameters for various domain sizes and see strong performance in all domains.

B. Utility Estimation

We used Pearson's correlation coefficients [10] to compare the differences between the actual utility and perceived utility of each negotiation in the final tournament.

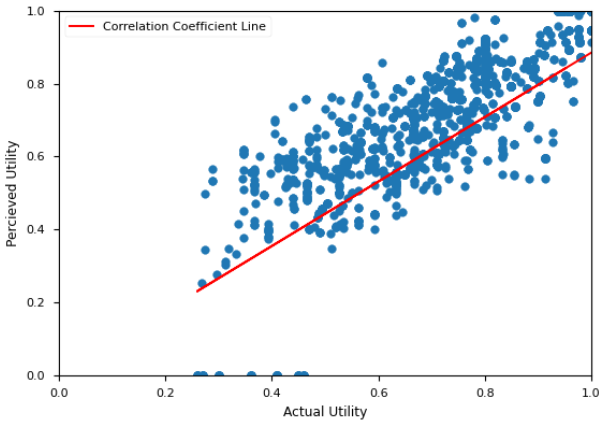

Fig. 6: Utility Correlation

The correlation coefficient produced a value of 0.88, indicating a strong positive relationship between the utility values, see Fig. 6. Although this has yielded positive results, there is still a margin of difference between what our agent perceives its utility to be.

VII. Future work

Given further time, we would have found some examples of larger datasets and run our hyperparameter tuning script using those domains. We believe that had we tuned the model to be more generalised across all domain sizes than Agent17 would have been one of the strongest agents in the tournament.

Further investigations into the Bayesian hyperparameter tuning could have involved testing it out with different optimisation functions, for example, using a weighted average of the NE and utility. An even more advanced approach to this would be to tune the model for different concession strategies, opponent models and preference elicitation techniques to produce a final model which was capable of switching between strategies and hyperparameters based on information about the domain.

For opponent modelling in particular, having access to a larger domain would have given us longer negotiation runs which would have allowed us to tune the Recent Bid Window. Furthermore, for opponent modelling, we would have also liked to pursue a heuristic method further by investigating a data-driven approach mentioned in Baarslag et al [4]. This would involve assessing the likely acceptance of a bid by matching on specific options for each issue alongside estimating opponent utility.

As we saw in the tournament results, we potentially overfitted our model to small domains. When constructing our agent we produced several opponent modelling methods and we considered using an ensemble technique to incorporate all of them with the hope that this would prevent any one of them overfitting. In future work, we would like to implement this technique to see if this would have the intended effect and we would see a more generalised agent.

VIII. Conclusion

Our work combined a number of different techniques and encapsulated them into Agent 17 effectively with the help of Bayesian hyperparameter Tuning. We saw from the evaluation section that it performed above average but also that it potentially overfitted to small domains. We were the strongest agent in the smallest domain and hence we have confidence that our approach is fruitful.

In the future work section we have suggested some ideas for how our implementation can be improved so that the agent sees similar levels of success on larger domains. We hypothesise that a model which is tuned on different domain sizes and is capable of changing its strategy based on the type of domain encountered would perform very well if this tournament was to be repeated.

We believe this this model demonstrated an impressive level of performance on all metrics, but ultimately we are encourage by the prospect of how our methods could be utilised to yield even stronger negotiation agents in the future.

References

[1] T. Baarslag, *Optimal non-adaptive concession strategies*, Jan. 1970. [Online]. Available: https : / / link . springer.com/chapter/10.1007/978-3-319-28243-5_9.

[2] K. Mansour, *A hybrid concession mechanism for negotiating software agents in ...* 2020. [Online]. Available: https : / / worldscientific . com / doi / abs / 10 . 1142 / S0218213020500165.

[3] O. Yucel, J. Hoffman, and S. Sen, "Jonny black: A mediating approach to multilateral negotiations," *Modern Approaches to Agent-based Complex Automated Negotiation*, pp. 231–238, 2017.

[4] T. Baarslag, M. J. Hendrikx, K. V. Hindriks, and C. M. Jonker, "Learning about the opponent in automated bilateral negotiation: A comprehensive survey of opponent modeling techniques," *Autonomous Agents and Multi-Agent Systems*, vol. 30, pp. 849–898, 2016.

[5] A. B. Yildirim, N. Sunman, and R. Aydoğan, "Ahbune agent: Winner of the eleventh international automated negotiating agent competition (anac 2020)," in *Recent Advances in Agent-Based Negotiation: Applications and Competition Challenges*, R. Hadfi, R. Aydoğan, T. Ito, and R. Arisaka, Eds., Singapore: Springer Nature Singapore, 2023, pp. 102–118, ISBN: 978-981-99-0561-4.

[6] D. Tsimpoukis, T. Baarslag, M. Kaisers, and N. G. Paterakis, "Automated negotiations under user preference uncertainty: A linear programming approach," in *Agreement Technologies*, M. Lujak, Ed., Cham: Springer International Publishing, 2019, pp. 115–129, ISBN: 978-3-030-17294-7.

[7] H. Nabli, "An overview on the simplex algorithm," *Applied Mathematics and Computation*, vol. 210, no. 2, pp. 479–489, 2009, ISSN: 0096-3003. DOI: https://doi.org/10.1016/j.amc.2009.01.013. [Online]. Available: https : / / www . sciencedirect . com / science / article / pii / S0096300309000198.

[8] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, no. null, pp. 281–305, Feb. 2012, ISSN: 1532-4435.

[9] X. Yufei, L. Chuanzhe, L. YuYing, and L. Nana, *A boosted decision tree approach using bayesian hyper-parameter optimization for credit scoring*, Feb. 2017. [Online]. Available: https : / / www . sciencedirect . com / science/article/abs/pii/S0957417417301008.

[10] P. Sedgwick, "Pearson's correlation coefficient," *BMJ*, vol. 345, e4483–e4483, Jul. 2012. DOI: 10.1136/bmj.e4483.