Multi-Agent Systems Report 2

Samuel Meyer (5648122) Sorin Dragan (6884393) Markos Polos (6943721)
Olusanmi Hundogan (6883273)

1 PEAS

1.1 Environment

The task consists of a **bilateral negotiation**. This is a process in which two parties make offers to each other in order to reach some mutual agreement.[5] The negotiation setting aims to organize a party. For each session, two agents with specific preferences negotiate a set of issues for the party. Within this **multi-issue domain**, two agents will bid to reach an agreement. The bilateral negotiation will use the **alternating offers protocol**. Bids and counter bids are placed in turns, creating a bidding history. The negotiation setting ends upon reaching a timeout, a maximum number of bidding turns, or a bilateral agreement.[2] The aim for each agent is to reach an agreement with the highest possible personal utility. The more the bid aligns with the agent's preferences, the higher its utility value. Without agreement, both agents get zero utility. Thus, agents are motivated to concede towards preferences of the other agent. However, the preferences are private and the agent needs methods adapted to **preference uncertainties**. Russel and Norvig define an environment in terms of a set of several characteristics.[8] In the following, we will discuss the agent environment per these characteristics. **Fully vs Partially observable:** The first characteristic to describe the environment refers to the observability of the environment. Our agent will not know the exact preferences of other agents. Because agents are not regarded as part of the environment, all the useful information is available to the agent. Therefore, we can refer to the environment as fully observable. **Single- vs Multiagent:** Every negotiation setting will require two agents interacting with each other. As our task environment requires negotiations, it must be regarded as a multiagent setting. **Competitive vs Cooperative:** The environment is a mixture between cooperative and non-cooperative. It is not cooperative in the game theoretic sense, as agents do not form a coalition to achieve a goal. However, the agents must somehow cooperate towards reaching an agreement that will benefit both of them. The game is also non-cooperative due to the fact that the negotiation setting focuses on each agent trying to push the decision towards his own personal preferences. Moreover, under particular strict settings and diametrically opposite preferences, the game can also take the form of a purely competitive game. **Deterministic vs Stochastic:** As established above, the only uncertainty that arises from the environment comes from the other agents with which we interact. As they are not part of the environment, it can be assumed to be deterministic. **Episodic vs. Sequential:** This environment not only provides information of the current situation but also bidding behaviors of the past. As an agent can form a bidding strategy based on the bidding history, the environment can be understood as sequential. **Static vs. Dynamic:** The environment changes only when an agent places a bid. Hence, the environment is static. **Discrete vs. Continuous:** This characteristic depends on how the bids will be placed at every negotiation setting. If the bids are placed in rounds, the environment will be discrete and if they are placed in real time the environment will be continuous.

1.2 Actuators

Actuators are used to act on the environment based on internal reasoning. The agent can either agree on a bid or make a counteroffer. Therefore, the only two possible actions are **bid** and **accept**. The bids provide certain utility values to each agent, depending on the agent preferences regarding a set of issues. The range of possible utilities of both agents forms the utility space. The utilities of a bid for each agent can be seen as a point in this utility space. Considering that an agent attempts to maximize its utility, the offers the agent makes and the offers it accepts are affected by the values of bids in the utility space. Once a bid has been accepted, this is considered and agreement in this space.

1.3 Sensors

Sensors are used to retrieve information from the environment for reasoning purposes. The agent can perceive the **bids** of other agents and its **personal utility** at each step, as we established that the game is played sequentially. Hence, we consider the bid history as an internal state of the agent, instead of a percept. At any point in the negotiation, the agent can also inform itself of the total and remaining amount of time left, which it can consequently use in its decision making strategies. Thus, we also consider **time** to be a percept. We further assume that the sensors are going to give us perfectly correct information about the bids and the time deadline.

1.4 Performance measure

We will use several baseline models to compare our model with: RandomWalker, HardHeaded, BRAM and Tit-for-tat.[3][9][6] These models represent random, boulware, conceder and adaptive strategies respectively that will elicit potential weaknesses and strengths of our agent. Thereafter, we devise a tournament in which our agent competes with other agents and rank it accordingly. There are multiple criteria on which this ranking could be based.[2] As the main performance measure, we decided on the utility sum, as an equivalent to the average utility, across all matches in the tournament. This better reflects the evaluation of the final tournament. For analysis purposes, we additionally consider the percentage of agreement, the distance to pareto frontier and robustness. As the negotiation sessions are neither discounted nor win-win oriented, we dismiss time of agreement, joint utility and distance to fair outcome. The agreement percentage will capture the amount of failed negotiations. The Pareto frontier is useful to capture the agent's ability to reach the optimal outcome, which also captures the maximal joint utility. The robustness will be quantified as the standard deviation of utility across all matches. All of these metrics will also provide insights into the individual contribution of each component in the BOA framework.

2 Agent Design Description

2.1 Acceptance Strategy

The acceptance strategy of our agent consists of three main ideas. The first is the time-based influence $\mathcal{E} \in (0, 1)$. For a received opponent bid bid_o^t at time t and the bid our agent will make one step later bid_a^{t+1}, the opponent bid will be rejected if $u(bid_a^{t+1}) > u(bid_o^t) + \mathcal{E}$. The value of $\mathcal{E}$ starts close to 0. However, to increase the odds of reaching an agreement by the end of the negotiation, the value of $\mathcal{E}$ increases as

the negotiation approaches the final timestep. The value of $\mathcal{E}$ is calculated with $x = \frac{timestep}{deadline}$ as function $f(x) = \frac{0.02}{1.05-x}$. The chosen function increases slowly for most the session and fast towards the end. If a bid has not been rejected as a result of the above, the second idea comes into play – the use of the opponent bidding history H. If the bidding history of an opponent has a bid with a higher utility for our agent than its current one, the bid is rejected and the next bid our agent proposes will be the past bid that was found in H. Thus, if facing an opponent that backs away from a concession, our agent attempts to hold them to that concession. After re-proposing the opponent bid, this bid is popped from H. This prevents the acceptance strategy from getting stuck re-proposing the same bid when an opponent does not ever move back towards the earlier concession. If the first two ideas do not lead to a rejection, the opponent bid is assessed based on the final notion. The final component is based on our agent's prediction of the next opponent bid. If our agent predicts that upon rejection, the next opponent bid will have a higher utility for our agent than the current bid, the current bid is rejected. If the opponent bid was not rejected through all three components, the bid is accepted. The estimation of the opponent's bidding strategy will be discussed in subsection 2.3. The full process is displayed in algorithm 1:

Algorithm 1: Acceptance Strategy

if $u(bid_a^{t+1}) \leq u(bid_o^t) + \mathcal{E}$ **:**
  if $\exists t' \in H$ *such that* $u(bid_o^t) < u(bid_o^{t'})$ **:**
    Reject bid, propose $bid_o^{t'}$ as next bid bid_a^{t+1}
    Pop $bid_o^{t'}$ from H
  else if $u(bid_o^t) \leq$ *predicted* $u(bid_o^{t+1})$ **:**
    Reject
  Accept bid
else:
  Reject bid

2.2 Bidding Strategy

The central idea behind our bidding strategy is a Monte Carlo Search Tree (MCTS) which was proposed by Buron et al. [4] They showed that the integration of the algorithm in the negotiation context outperformed the tit-for-tat strategy. The same method proved to be successful in sequential games with large branching factors like Go.[7] Furthermore, the method is still not fully explored in the context of negotiation and we assume a lack of obvious counter strategies. The method aims to find the best bid using simulation and a tree structure. The simulation improves with the accuracy of the opponent bidding strategy model. Besides the MCST approach, we also propose a Self-Balancing Binary Tree approach as a fallback if the prediction error remains too high. **Monte Carlo Tree Search:** The root of the tree will represent the initial state of the game, in which no bid has been offered yet. Each state (node) of the game will keep information about its score (utility) and number of visits. The first bid of the agent will be a random bid with utility higher than 0.9. A **node expansion** will be performed according to a progressive widening. Meaning, if the number of visits of the parent node is greater or equal to the number of children nodes, a new relevant bid in the same space will be picked. The expansion could consist of a random bid among possible ones. However, an important observation of the early stages of the game is that both the opponent bidding model and the corresponding

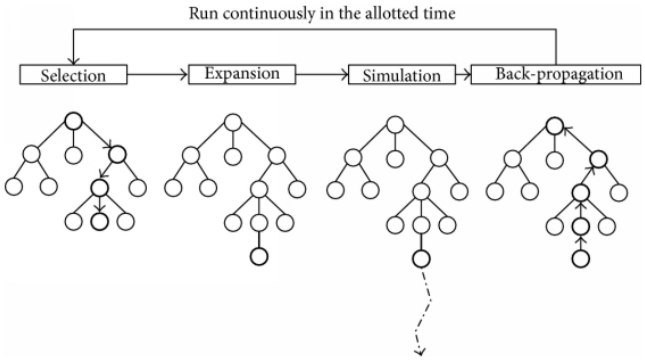

Figure 1: The steps of the Monte Carlo Tree Search algorithm.[10]

bidding strategy of the MCTS will be poor. Thus, we consider trying to expand using a **TIT-FOR-TAT** strategy instead of a random one. A **rollout** (simulation) will be performed using the opponent strategy model. We will consider that a leaf node is reached based on our acceptance strategy. The opponent acceptance strategy will not be used as the current approach simulates worst-case scenarios. Hence, a premature acceptance of the opponent will be beneficial. After a leaf node is reached, the **back-propagation** step will update the nodes on the upper levels with the obtained score. **Nodes selection** will be performed based on a modified version of the UCB1 formula present in Buron et al. **Self-Balancing Binary Tree:** If the model for the opponent bidding strategy a displays a high prediction error (see subsection 2.3), we consider a simpler strategy. The first step of this strategy will take all the possible bids (which is less than 4000 in the party domain) and construct a **self-balancing binary tree** using the utility calculated from each bid. Therefore, each node of the tree will represent a utility score and a list of combinations that result in that utility score. Once we have this tree, we can start from the bottom-right leaves (the high utility) and bid progressively making concessions that slowly go up the tree approaching the root, but never going on the left side of the tree. Assuming a normal distribution of utility scores, the root should have a utility value close to 0.5.

2.3 Opponent Bidding Strategy Model

A core component of MCTS is the simulation of various session rollouts. However, learning the opponents behaviour which can suddenly change is a non-trivial moving target problem.[2] Baarslag et. al mentions several methods for learning the opponents bidding strategy.[2] Generally divided into regression analysis and time series analysis, the latter do not assume any underlying decision function. In particular, the signal processing method Gaussian Process Regression has multiple advantages. First, it does not require pre-training. Second, gaussian processes can be used for classification problems. Third, it can predict moving targets. The aim of using GPR in this context is to estimate the opponents expected bid y_* and variance σ_*^2 given an unobserved x_* and an observed history of observation and opponent bid tuples (X, Y). Both, y_* and σ, can be used to sample opponent bids in a simulation. Equations 1 and 2 describe the calculation.[4]

$$K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_n, x_1) \\ \vdots & \ddots & \vdots \\ k(x_1, x_n) & \cdots & k(x_n, x_n) \end{bmatrix} \qquad K_* = [k(x_*, x_1), \cdots, k(x_*, x_n)] \tag{1}$$

$$y_* = K_* K^{-1} Y \qquad \sigma_*^2 = K_{**} - K_* K^{-1} K_*^T \tag{2}$$

Equation (1) describes the proximity in terms of covariance matrices K and K_* for features X and x_*. Typically, X reflects the time dimension alone. However, we will also consider our previous bids, opponent preferences or a combination. K refers to the covariance matrix within the set of X, whereas K_* describes the covariance vector between x_* and X. $k(a, b)$ denotes a kernel function as proximity measure. With Y being the previous bids and y_* the expected bid, we can calculate $\mathbb{E}[y_*|K_*, K, Y]$ and its variance σ_*^2. By assuming that each X form a multivariate Gaussian, equations in (2) follow. We choose RQF as kernel function, as it promises the best performance according to Buron et al.[4] In the case of a GPR, the kernel function is used to construct the covariance matrix. As proposed by Baarslag et. al., we will use the mean squared error (MSE) to determine the prediction error.[2] This will guide the use of the prediction or switch to a fallback bidding strategy. As a further note, we will make the simplifying assumption that the each issue does not interact with another. Hence, we can predict the bid by predicting each issue separately.

2.4 Opponent Preference Model

First, it should be noted that according to Tim Baarslag the opponent model makes a surprisingly small contribution to the performance of a negotiation strategy.[1] However, as the HardHeaded model in ANAC 2011 showed, it had a significant impact on the outcome of the tournament and makes it reasonable to implement. Given the small yet crucial contribution, we will implement a frequency-based opponent preference model instead of a more sophisticated method such as bayesian learning. Figure 2 shows how frequencies within the opponent's bidding history can offer insight to its preferences. We can use these preferences as an additional input for the opponent bidding strategy model and as a way to estimate the Pareto frontier. Additionally, two assumptions can facilitate the preference modeling. First, the earlier bids will reflect the opponent's preference more than the later ones. Second, the opponent will concede more on issues with less importance than on issues with high importance.[2] Thanks to the nature of the frequency-based model, these assumptions can be incorporated in the form of time-discounted frequencies or variance statistics respectively.

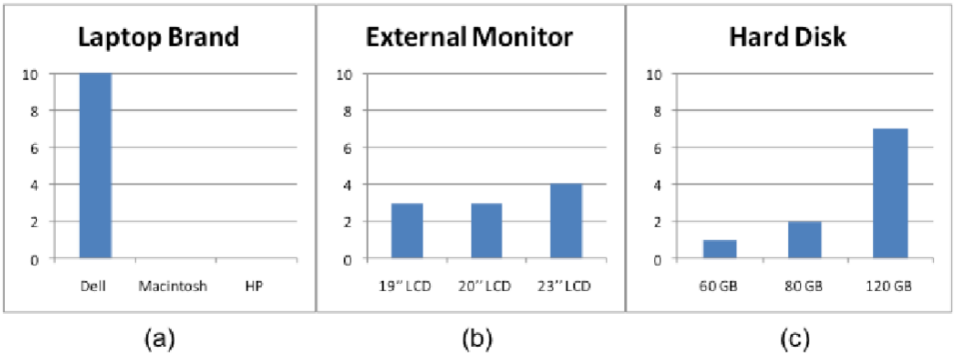

Figure 2: A histogram example with the last 10 offers made by the opponent agent in the Laptop domain taken from Fishel et. al.[6]

References

[1] Tim Baarslag, Alexander Dirkzwager, Koen Hindriks, and Catholijn Jonker. The significance of bidding, accepting and opponent modeling in automated negotiation. volume 263, 08 2014.

[2] Tim Baarslag, Mark J. C. 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, 2016.

[3] Tim Baarslag, Koen Hindriks, and Catholijn Jonker. *A Tit for Tat Negotiation Strategy for Real-Time Bilateral Negotiations*, page 229–233. Studies in Computational Intelligence. Springer, 2013.

[4] Cédric L. R. Buron, Zahia Guessoum, and Sylvain Ductor. Mcts-based automated negotiation agent. In Matteo Baldoni, Mehdi Dastani, Beishui Liao, Yuko Sakurai, and Rym Zalila Wenkstern, editors, *PRIMA 2019: Principles and Practice of Multi-Agent Systems*, Lecture Notes in Computer Science, page 186–201. Springer International Publishing, 2019.

[5] Juan Ramón Rabuñal Dopico, Julian Dorado, and Alejandro Pazos. Encyclopedia of artificial intelligence (3 volumes), 2009.

[6] Radmila Fishel, Maya Bercovitch, and Ya'akov (Kobi) Gal. *BRAM Agent*, pages 213–216. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[7] David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.

[8] Russell Stuart and Norvig Peter. *Artificial intelligence-a modern approach 3rd ed*. Berkeley, 2016.

[9] Thijs van Krimpen, Daphne Looije, and Siamak Hajizadeh. *HardHeaded*, page 223–227. Studies in Computational Intelligence. Springer, 2013.

[10] Maciej Świechowski, HyunSoo Park, Jacek Mańdziuk, and Kyung-Joong Kim. Recent advances in general game playing. *TheScientificWorldJournal*, 2015:986262, 09 2015.