Terminator2020: A Modified Approach Using Time Dependent Tacticts

Onkar Dhavan
32199473
ord1n20@soton.ac.uk

Group 21

Shruti Gaikwad
32199481
sg4n20@soton.ac.uk

Marco Edoardo Palma
29371481
mep2g17@soton.ac.uk

Srikanth Srivenkata
32176929
ss22g20@soton.ac.uk

1 INTRODUCTION

The coursework aimed to produce an autonomous agent to compete in an automated agent's league, against opponents in a variety of scenarios. The league involves a closed setting and preference uncertainty enabled and consequently the developed agent has three distinct parts, namely user modelling, opponent modelling and the bidding and accepting strategy. This document briefly describes the development process and thoroughly explains the components and working of the final agent.

2 THE 'BASELINE' AGENT

A basic agent with acceptable performance (baseline agent) was built before developing the final agent. This agent employed the strategy of sending out the same or random bids to avoid the opponent from modelling our agent correctly. This strategy was used by a few top performing agents in the ANAC (Automated Negotiating Agents Competition) 2019 competition and hence was used. The agent had no opponent and user modelling and simply accepted any bids above 0.8 utility before 90% of the time was up. After 90% of the time was over, the agent accepted bids over 0.5 utility. The goal was to make a final agent to at least outperform this basic negotiating strategy.

3 STRATEGY

3.1 Parameters

For creating a bidding strategy, the agent uses these parameters:

(1) User model: - The provided Preference Uncertainty is utilised for the estimation of a user model though a custom implementation of Cuckoo Search Optimization [2] (Section 3.3).
(2) Conceding Value: - Conceding value is a minimum utility above which our agent is selecting bids for the opponent. Conceding value is calculated by using the Time Ratio and Exponential Variable. The Time Ratio used as a constraint for selecting minimum Conceding Value based upon the ratio of the current time and total time. Conceding Value is not a constant value, it always changes concerning time. The minimum utility based on time is selected here to give as an input to the Exponential Variable function. The minimum utility and Time Ratio constraints are selected after experimentation.
(3) Exponential Variable: - The Exponential Variable uses Boulware Time-Dependent tactics [5] to exponentially calculate the final Conceding Value using the minimum utility, maximum utility, current time, and the total time.

3.2 Initialization

During the initialization stage, the agent employs a custom implementation of Cuckoo Search Optimization (Section 3.3), to attempt the derivation of a user model solely utilizing the partial preference ordering given at the start of the negotiation [2].

3.3 User Modelling

Cuckoo Search Optimisation (CSO) [1] was implemented for the derivation of user models, given some partial preference ordering [2].
CSO aims at simulating the cuckoo bird's parasitic approach to egg-laying, upon which its reproduction strategy heavily relies, to iteratively improve an initial set of solutions (generations, nests), each of which quality is heuristically estimated. In the contest of user modelling, a cuckoo searches for the best nest to parasitise, as an agent searches for the best performing user model.

A candidate user model is a set of free weights of which absolute value $\in [0, 1]$. Each of these weights is logically mapped to an option for a given issue in the negation domain, in a fashion such that for each option, a weight is bounded. Hence, given a bid, the estimated user utility is the weighted sum of all option weights selected by the bid:

$$U(B, M) = \sum_{i \in issues(B)} w_i \times M_{(i,option(B,i))} \tag{1}$$

where: B is the bid, M the user model, $issues$ is a function which given a bid, returns its set of issues $issues : B \rightarrow \{Issue\}$, $option$ is a function that given a bid and an issue returns the option number selected by in the bid for that issue $option : B \times I \rightarrow Z$, $M_{(i,option(B,i))}$ represent the weight the user model assigns for that issue and option, and w_i is a weight normalising the value of the issue i computed as:

$$w_i = \frac{argmax \forall_{o \in i} M_{(i,o))}}{\sum_{i \in Domain} argmax \forall_{o \in i} M_{(i,o))}} \tag{2}$$

The search begins with a set of user models with randomly generated option weights, which represents the first generation of solutions. These are then ranked according to their closeness in representing the partial preference ordering given at the negotiation start. The preference order is derived from a user model, by sorting each bid by its estimated user utility (U). Ties are resolved

by ranking each tying subset with the same rank value, equal to the average of their ranks (had no tie occurred). This ranking strategy hence allows for Spearman's Rank Correlation [2] to be utilised in scoring the closeness of the model's generated rank, with the given partial rank.

From each generation, the best performing model is drawn and utilised by all other models to produce a new candidate 'child' model by performing a Lévy Flight about their option weights. Hence, for each user model M, and the best performing model M_b, a new model M_n is generated with weights:

$$\forall_{w \in M} M_{n_w} = toRange(M_w + (M_w - M_{b_w}) \times (r_1 \times 0.01 \times \frac{r_2 \times \sigma_U}{|r_3 \times \sigma_V|^{\frac{1}{\beta}}})) \quad (3)$$

where: s is an option weight; *toRange* maps values < 0 to 0, values > 1 to 1, and any other value to itself; r_i are newly generated random values $\in [0, 1]$; σ_U, σ_U and β are Lévy Flight constants mapped to 0.6966, 1, and 1.5 respectively. Hence, the newly generated user model M_n replaces its corresponding 'father' model M iff its Spearman's rank correlation score is greater. Finally, for each user model M and two other randomly chosen models M_1 and M_2, a new candidate child user model M_m is generated with weights:

$$\forall_{w \in M} M_{m_w} = \begin{cases} M_w, & \text{if } r_1 \leq 0.25 \\ toRange(M_w + (r_2 \times (M_{1_w} - M_{2_w}))), & \text{otherwise} \end{cases} \quad (4)$$

Similarly to the previous phase, M_m replaced the 'father' model if and only if its Spearman's rank correlation score is greater.

The GENIUS framework was utilised for testing this approach to user modelling for a variety of domains and user preference orderings. In our testing, the population size was capped at 100, whilst generations were run in groups of 20, utilising the Spearman's rank correlation score of the best performing user model stabilised for 3 consecutive rounds. Whilst the final user model was derived utilising solely the partial preference ordering provided upon initialisation of the negotiation, this was ultimately scored against the complete preference ordering of the all bids derivable from the domain. This process was repeated for increasing numbers of bids provided in the initial ordering.

Considering the results obtained with regards to a moderate domain such as the Part-domain found in ANAC, with a total bid count of 3072, this implementations of CSO for user modelling, constantly produced an average Spearman's rank score of above 0.70, for a partial order to total order ratio greater of 0.0015 (ie. 4 - 5 bids) with regards to user profile 1, whilst higher accuracy was obtained with regards to user profile 5, which yield a score of above 0.90 once at least 4-5 bids were given in the preference order. These results go some way to summarising the behaviour of this approach. In fact, for a small number of partial orderings (in this domain 2-3), CSO found too quickly too many options-weight combinations which all lead to a perfect Spearman's score, hurting the evolution of the search. This behaviour is especially highlighted by the uniformity of the user preference profile: higher uniformity leads to higher uncertainty in generalising the partial order. Hence the accuracy difference between user profile 1 and 5, with the former yielding greater preference uniformity across all issues, compared to the latter. As hinted, this behaviour remained consistent with domains with both smaller (such as ANAC's Laptop-domain, with size 32) and larger domain sizes (such as ANAC's Windfarm-domain, with size 7056), with the search typically requiring a larger number of generations for larger domains. Figure [1].

3.4 Opponent Modelling

A revised version of the classic frequency modelling approach was used for opponent modelling [3]. First, A HashMap was created to store the frequencies of issue outcomes from all received bids and was updated after each bid. The bids were divided into windows.The window size was set to 10 and the weights were updated once every new window of bids fills up. Four more hash-maps were created to record the frequencies and valuations of adjacent windows. The valuation function used is defined as (5).

$$\hat{V}_i(j) = \frac{(1 + \sum_{o \in O_{1 \to t}} \delta_i(j, o))^\gamma}{max_{k \in AT}(1 + \sum_{o \in O_{1 \to t}} \delta_i(k, o))^\gamma} \quad (5)$$

where δ_i (j, o) is 1 if the value j is used for issue i in offer o and 0 otherwise. AT = 1, 2, . . . , n are the negotiation issues. It is to be noted that the frequency is smoothed using Laplace (add-one) smoothing. The valuations are computed using the two frequency hash-maps of adjacent bid windows. The agents' preferences are represented by means of linear additive utility functions in the form of (6).

$$U(o) = \sum_{i \in AT} w_i * V_i(o_i) \quad (6)$$

where w_i represents the importance of the negotiation issue i, o_i represents the value for issue i in offer o, and $V_i(.)$ is the valuation function for issue i, which returns the desirability of the issue value. Before evaluating the utilities, the frequency distributions of the outcomes of issues of the current and the next window bids are put through the Chi-Squared test. If the outcome is over 0.05, this issue is added to the 'e' issue array. If the threshold is not met, the Expected utilities are calculated and if the new expected utility is lower than the previous value, then the concession variable is set to True. For each issue in the 'e' array, weights are updated using the formula (7).

$$\Delta(t) = \alpha * (1 - t^\beta) \quad (7)$$

The frequency of a negotiation value j of issue i in a window of offers O is calculated as in (8).

$$Fr_i(j, O) = \frac{1 + \sum_{o \in O} \delta_i(j, o)}{n + |O|} \quad (8)$$

The pseudo-code of the algorithm has been described in Algorithm 1. The weight update pseudo-code has been shown in Algorithm 2. α was set to 10 and β to 5. After every 10 (window size) bids, the hash-maps of the previous window is replaced with the hash-maps of the current window, whilst the hash-maps of the current window are initialized to zeroes. Finally, A 'getUtility' function was made available for the returning of the opponent's expected utility.

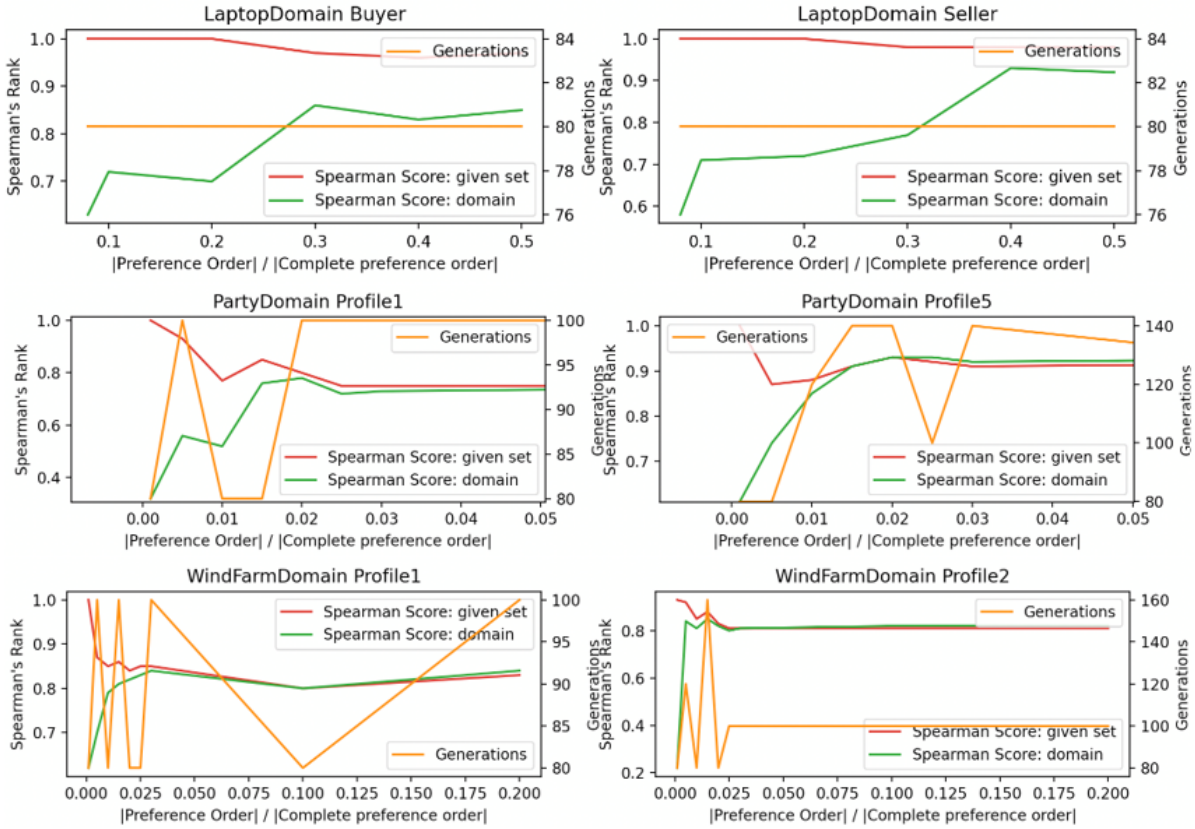

Figure 1: Sample results CSO for user modelling.

Algorithm 1: Opponent Modelling Pseudo code

```
e ← ∅
concession ← False
foreach i ∈ N do
    w_i ← w_i'
end
foreach i ∈ AT do
    F_i' ← (Fr_i(1, O'), ..., Fr_i(n, O'))
    F_i ← (Fr_i(1, O), ..., Fr_i(n, O))
    if p_val > 0.05 then
        e ← e ∪ i
    else
        V_i ← (V̂(1), ..., V̂(n))
        E[U_i(O')] ← V_i * F_i'
        E[U_i(O)] ← V_i * F_i
        if E[U_i(O')] < E[U_i(O)] then
            concession ← True
        end
    end
end
```

Algorithm 2: Weight update Pseudo code

```
if ||e|| ≠ n and concession = True then
    foreach i ∈ e do
        w_i ← w_i' + Δ(t)
```

3.5 Accepting Strategy

For the first 20% of the time, our agent accepts the offer only if the utility of last received bid is greater than 0.75. After that, till the remaining time is greater than 90%, our agent accepts the bid only if the last received bid gives the utility of 0.75 to our agent otherwise our agent offers the bid calculated using the User Modelling (Sec. 3.3). If the time remaining is less than 10% of the total time and the utility of last received bid is greater than 0.65, our agent accepts the last received bid. If the remaining time is less than 99%, our agent receives a bid with any utility. By using this accepting strategy, our agent gets a little higher utility than the opponent. It does not reduce the social welfare as the bidding strategy is designed to get a little higher utility for our agent than the opponent. The values used as a constrained are included after the experimenting with the different values.

Table 1: Tournament results

Agent number	Average distance to Nash point	Average utility	Acceptance fraction	Overall score
1	0.130352	0.825641	1	0.7180
2	0.244858	0.820308	0.9	0.5575
3	0.128584	0.828038	1	0.7215
4	0.130864	0.811611	1	0.6888

3.6 Bidding Strategy

In bidding strategy, for the first 20% of the time, our agent gives the bid having a utility between 0.90 and 0.95. The agent then proceeds by generating random bids using Conceding Value (Sect. 3.1.2) as a constraint. To calculate Conceding Value, Time Ratio is needed. Depend on the Time Ratio, the minimum utility changes. The current time and minimum utility are used as a parameter to calculate Exponential Variable (Sect. 3.1.3) along with maximum utility and the total time. The Exponential Variable then calculates the value using the above parameters, using Boulware time-dependent tactics [5].

When computing a counteroffer, the agent aims at making a negotiation aware concession, ensuring the user preference (if any) is a voting member in the selection of the counteroffer, as well as not following a clear direction in concession process, so to reduce the chances of opponents learning the agent's preferences. Hence, 50 bids are generated by Exponential Variable strategy. The agent then employs the user model generated through CSO during the initialization phase, to extract the bid which is expected to be the most preferred by the user; resolving ties by selecting the first tying bid found. However, the agent will revert to returning a random bid of these, if the user model is unavailable. This may occur if the agent was not provided with preference ordering during initialization, or if the confidence about the user model is too low, due to either: the ratio of bids included in the initial preference ordering and the total number of possible bids being less than 0.015 (Section 3.3), or the model's Spearman's rank score is less than 0.5.

4 EVALUATION

Four versions of the agent were developed.

(1) Agent with only the bidding strategy
(2) Agent with User and Opponent Modelling
(3) Agent with User Modelling
(4) Baseline Agent

All the agents were put through the same tournaments. The agents competed against three popular agents from previous competitions. 200 was set as the round limit. Stacked Alternating Offers Protocol was used with six bidding profiles. The overall score was calculated as:

$$Overallscore = (1 - distance\ to\ Nash\ point) \times utility \times \frac{agreements}{negotiations} \quad (9)$$

The results were tabulated and the average distance to Nash point, utility and the fraction of bids accepted were computed and tabulated in Table 1

5 CRITICAL ANALYSIS AND FUTURE WORK

It was inferred from these results that the agent with user modelling performed the best. Whereas although the agent with opponent modelling achieved a similar average utility, it yields greater average distance from Nash point, and only 90% of the bids lead to acceptance. The reason behind this drop in performance was due to the opponent model and the bidding strategy non-coordinated operation. The bidding strategy merely focused on increasing its acceptance utility, not fully exploiting the opponent's modelling information upon counter-offering. To the contrary, user modelling yield better results, as it reinforced the bidding strategy ultimate goal, hence providing concessions which attempted to maximise the user's utility. Ergo, despite having a strong choice of individual methods, the combination of all ended up degrading the agent's performance. Hence, the designing of a bidding strategy better suited to the usage of the opponent modelling is required; this may be achieved by focusing on social welfare aspect of the negotiation, rather than solely on the agent's utility.

Word count: 2426

REFERENCES

[1] Yang, X.S. and Deb, S., 2009, December. Cuckoo search via Lévy flights. In 2009 World congress on nature & biologically inspired computing (NaBIC) (pp. 210-214). IEEE.
[2] Bagga, P., Paoletti, N. and Stathis, K., 2020. Learnable Strategies for Bilateral Agent Negotiation over Multiple Issues. arXiv preprint arXiv:2009.08302.
[3] Tunal O, Aydoan R, Sanchez-Anguix V. Rethinking frequency opponent modeling in automated negotiation. In: International Conference on Principles and Practice of Multi-Agent Systems; Nice, France; 2017. pp. 263-279.
[4] 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.
[5] "Tactics and Strategies/ Time-dependent Tactics", [Online]. Available: https://web.fe.up.pt/ eol/schaefer/diplom/TacticsAndStrategies.htm