Baseline Strategies for the ANAC Automated Negotiation League

Kotone Ninagawa
Brown University
Providence, RI, USA
kotone_ninagawa@brown.edu

Yasser Mohammad
NEC Datascience Research Laboratories
Tokyo, Japan
y.mohammad@nec.com

Amy Greenwald
Brown University
Providence, RI, USA
amy_greenwald@brown.edu

ABSTRACT

We present *agentKT* (agentKiTsune), a finalist in the 2020 Automated Negotiation League (ANL) of the Automated Negotiating Agents Competition (ANAC). Specifically, we describe our agent's elicitation and negotiation strategies, as well as our approach to learning the user's and opponent's utility functions. The primary purpose of this paper is to identify core subproblems inherent in automated negotiation—specifically, ANL—and to provide heuristic solutions to these subproblems that can serve as baselines in future runs of the competition. Indeed, the ANL organizers are planning to incorporate our code modules into an upcoming release of their platform for exactly this reason—so that future developers can benchmark their proposed solutions against a community standard. This paper is intended to accompany those code modules, by providing an English description of these baselines. We hope this paper will also serve to encourage more AAMAS researchers to participate in future renditions of ANAC by easing the barrier to entry, and by clarifying what progress in automated negotiation means, as something akin to increasing accuracy on a supervised learning task.

KEYWORDS

Automated bilateral negotiation, preference uncertainty, learning utility functions

ACM Reference Format:
Kotone Ninagawa, Yasser Mohammad, and Amy Greenwald. 2021. Baseline Strategies for the ANAC Automated Negotiation League. In *Appears at the 3rd Games, Agents, and Incentives Workshop (GAIW 2021). Held as part of the Workshops at the 20th International Conference on Autonomous Agents and Multiagent Systems., London, UK, May 2021*, IFAAMAS, 9 pages.

1 INTRODUCTION

Negotiation is one of the most prevalent methods for reaching agreements between self-interested parties. Negotiation research can be traced back to Nash's seminal work on bargaining theory [8], and Rubinstein's analysis of the alternating offers protocol in the perfect-information case [10], both major game-theoretic advances. More recently, the study of *automated* negotiation has attracted researchers in multi-agent systems (e.g., [5]) and machine learning (e.g., [6]). In automated negotiation, autonomous (software) agents negotiate among themselves, or with human negotiators, on behalf of their users.

The Automated Negotiation Agent Competition (ANAC) is an international tournament that serves as a benchmark for evaluating automated negotiation strategies. In the 2020 Automated Negotiation League (ANL), in particular, the challenge was "to design a negotiating agent that can elicit preference information from a user during a negotiation." In other words, agents represent a user whose utility function it does not know. The agent is instead given a partial ranking of outcomes, based on which it can infer its user's utility function. Additionally, it may elicit further information about the utility function by querying the user and incurring an elicitation cost. On the implementation side, agents run two parallel threads; one is used to place elicitation queries, and the other, to negotiate. This way, an agent can elicit information from its user while simultaneously negotiating with its negotiation partner, also known as its opponent.

The negotiation mechanism adopted by ANL is a variant of Rubinstein's alternating offers protocol [10] called the Stacked Alternating Offers Protocol [2]. It involves two agents, who take turns making offers for a finite number of rounds and/or seconds. One agent opens the negotiation with an offer, after which the other agent takes one of the following actions:

(1) Accepts the offer
(2) Responds with a counteroffer, thus rejecting and overriding the previous offer
(3) Walks away, thus declaring an end to the negotiation, without having reached an agreement

This process repeats until either an agreement or a deadline is reached. To reach an agreement, both parties must accept the offer. If no agreement is reached by the deadline, the negotiation fails.

While a game-theoretic analysis of this negotiation mechanism might prescribe a reasonable strategy for an agent, carrying out such an analysis is highly non-trivial. Not only is it an incomplete-information game, in the sense that knowledge of the opponent's utility function is limited, the agent does not even have complete knowledge of its own (i.e., its user's) utility function. Indeed, deriving the game-theoretic equilibria of this negotiation game is a wide open problem.

As an alternative heuristic, yet tractable, approach, we identify three core subproblems this game comprises, and present baseline solutions to each one, which together formed *agentKT* (agentKiTsune), a finalist in ANL 2020. The three core subproblems are: 1. *user modelling*: eliciting the user's preferences in the form of a partial ranking of outcomes, and learning the user's utility function from this partial ranking; 2. *opponent modelling*: inferring the opponent's

Appears at the 3rd Games, Agents, and Incentives Workshop (GAIW 2021). Held as part of the Workshops at the 20th International Conference on Autonomous Agents and Multiagent Systems., Aziz, Ceppi, Dickerson, Hosseini, Lev, Mattei, McElfresh, Zick (chairs), May 2021, London, UK. © 2021 Copyright held by the owner/author(s). …$ACM ISBN 978-x-xxxx-xxxx-x/YY/MM

negotiation strategy and its utility function from its observed behavior, namely the offers it places; and 3. developing an empirically robust *negotiation strategy*, both for placing and accepting offers.

To solve the user modelling problem, our agent's elicitation method places carefully constructed queries, each of which generates comparisons across multiple outcomes. The user's utility function is then learned by converting these comparison data into a regression problem. We do not attempt to model the opponent's negotiation strategy, as that would be exceedingly difficult in recent renditions of ANL, because negotiations are short, and the rules prohibit learning across negotiations. Instead, we model only the opponent's utility function, based on the opponent's offers. Finally, our agent's negotiation strategy, which relies on our estimates of the user's and the opponent's utility functions, is, like most ANAC agents', time-dependent [5]. Specifically, it incorporates ideas used by *Atlas3* [12], the winning agent in ANAC 2015.

The purpose of this paper is to formally define the core subproblems that *agentKT* solves, and to describe its heuristic solutions, so that they can serve as baselines by which to gauge research progress in future runs of the competition. Indeed, the ANL organizers are planning to incorporate our code modules into an upcoming release of their platform for exactly this reason—so that future developers can benchmark their proposed solutions against a community standard. This paper is intended to accompany those code modules, by providing an English description of these baselines.

This paper is organized as follows. In Section 2, we describe our agent's method of inferring its user's utility function. Section 3 then outlines our agent's approach to opponent modelling. We next present our agent's elicitation method (Section 4). Section 5 describes our agent's offering and acceptance strategies. Our acceptance strategy is not original; we did our best to re-implement the strategy of the winning agent from 2015 [12], and present here an English description of our understanding of their strategy, as none was available previously, to our knowledge. In Section 6, we present a cursory evaluation of our agent's overall performance. Demonstrating our agent's overall success is not our primary goal; establishing baselines is. Section 7 outlines the ANAC 2020 competition results.

2 INFERRING THE USER'S UTILITY FUNCTION

In ANL 2020, agents do not know the user's utility function, but are instead given a partial ranking of the user's preferred outcomes. This partial ranking always includes two labelled outcomes, ω_{worst} and ω_{best}, whose utilities are minimal and maximal, and whose values we assume to be 0 and 1, respectively. Here, we describe how our agent infers its user's utility function given this information.

Problem Statement. Let Ω denote the space of outcomes, with notable outcomes ω_{worst} and ω_{best}, and let $f : \Omega \to \mathbb{R}$ denote the user's utility function. Given a small, but non-empty, set of comparison data $C = \{\omega_i \geq \omega_i' \mid i = 1, \ldots, m\}$ that includes the agent's least and most preferred outcomes, where a pair $\omega_i \geq \omega_i'$ indicates that the user prefers outcome ω_i to outcome ω_i', we aim to infer an estimate $\hat{f}$ of the user's true utility function f. □

Let K denote the total number of issues in a negotiation domain. For issue $k \in \{1, \ldots, K\}$, let V_k denote the set of possible values, with $|V_k| = J_k$. For example, if the negotiation domain is "party," issue 1 is "food for a party," and the possible food items are pizza, sushi, pasta, and sandwiches, then $V_1 = \{\text{pizza, sushi, pasta, sandwiches}\}$ and $J_1 = 4$. We also let $\pi_k(\omega) \in V_k$ be a selection function that returns the value of issue k under outcome ω. Continuing with our example, if outcome ω's value for issue 1 is sushi, then $\pi_1(\omega) =$ sushi.

In ANL 2020, the agent is also told that its user's utility function f has linear structure. For issue $k \in \{1, \ldots, K\}$, let $f_k : V_k \to [0, 1]$ be a function that maps issue k's values to utilities. Then f is defined as a convex combination of f_k's, weighted by θ_k's, where $\theta_k \in [0, 1]$ and $\sum_{k=1}^{K} \theta_k = 1$. That is, $f(\omega) = \sum_{k=1}^{K} \theta_k f_k(\pi_k(\omega))$. *AgentKT* estimates the requisite weights using a heuristic by which it predicts the relative importance of the issues to the user.

If the K issues are sorted in ascending order of importance (from least important to most important) such that $\sigma(k)$ denotes the importance of issue k (i.e., its rank in this list), we set

$$\hat{\theta}_k = \frac{1}{2K} + \frac{\sigma(k) - 1}{K(K-1)} \tag{1}$$

This procedure amounts to setting $\hat{\theta}_{\sigma^{-1}(1)} = \frac{1}{2K}$ (for the least important issue) and $\hat{\theta}_{\sigma^{-1}(K)} = \frac{3}{2K}$ (for the most important issue), with the remaining $\hat{\theta}_k$'s taking values in between, in uniform increments. Observe that $\hat{\theta}_k \in [0, 1]$ and $\sum_{k=1}^{K} \hat{\theta}_k = 1$.

We explain how we set the order of importance of the issues for our user and for the opponent in Sections 4 and 3, respectively. Given such an ordering, Figure 1 presents an example of the corresponding θ_k's.

issues	food	drinks	dessert	location	music
$\sigma(k)$	1	2	3	4	5
θ_k	0.10	0.15	0.20	0.25	0.30

Figure 1: Example θ_k's when $K = 5$. The order of importance of the 5 issues, from least to most, is food, drinks, dessert, location, music.

Regression. Given a set of m pairwise comparisons, we seek functions $\hat{f}_k$ such that $\hat{f}$ is consistent with the given comparisons. To find such $\hat{f}_k$, we associate the ith comparison $\omega_i \geq \omega_i'$ with the constraint $\hat{f}(\omega_i) \geq \hat{f}(\omega_i')$. We then use the COBYLA optimization method [9] to minimize the sum of the utility differences of consecutive outcomes in the partial ranking.

More formally, given θ_k, for all $k \in \{1, \ldots, K\}$, and given the outcomes ω_{worst} (ω_{best}) whose utilities are minimal (maximal),

$$\min_{\hat{f}_k \in \mathbb{R}^{J_k}, k \in \{1,\ldots,K\}; \epsilon_i \geq 0, i \in \{1,\ldots,m\}} \sum_{i=1}^{m} \epsilon_i \tag{2}$$

s.t.

$$\hat{f}(\omega_i) + \epsilon_i \geq \hat{f}(\omega'_i), \qquad \forall \omega_i \geq \omega'_i, \forall \epsilon_i \geq 0, i \in [m] \quad (3)$$

$$\hat{f}(\omega_i) = \sum_{k=1}^{K} \theta_k \hat{f}_k(\pi_k(\omega_i)), \qquad \forall i \in [m] \quad (4)$$

$$\hat{f}_k(\pi_k(\omega_{\text{worst}})) = 0, \qquad \forall k \in \{1, \ldots, K\} \quad (5)$$

$$\hat{f}_k(\pi_k(\omega_{\text{best}})) = 1, \qquad \forall k \in \{1, \ldots, K\} \quad (6)$$

The COBYLA optimization package allows the developer to initialize the decision variables based on their domain knowledge, to aid the solver. We did not avail ourselves of this opportunity; we simply initialized the $\hat{f}_k$'s uniformly at random.

3 INFERRING THE OPPONENT'S UTILITY FUNCTION

While the structure of the opponent's utility function f' is known to be identical to our own agent's: i.e., $f'(\omega) = \sum_{k=1}^{K} \theta'_k f'_k(\pi_k(\omega))$, inferring the opponent's utility function is harder than inferring the user's for two reasons: first, the opponent's best and worst outcomes are unknown, and second, the agent is not given a partial ranking of the opponent's preferred outcomes.

Problem Statement. Let Ω denote the space of outcomes, and let $f : \Omega \to \mathbb{R}$ denote the user's utility function. Given an initially empty set of comparison data $C = \{\omega_i \geq \omega'_i \mid i = 1, \ldots, m\}$, where a pair $\omega_i \geq \omega'_i$ indicates that the user prefers outcome ω_i to outcome ω'_i, we aim to infer an estimate $\hat{f}$ of the opponent's true utility function f. □

As we do not know the opponent's best and worst outcomes, we cannot immediately apply the aforementioned regression solution that we use to infer the user's utility function. Instead, *agentKT* builds its estimate of the opponent's utility function using the information contained in the outcomes the opponent offers during each round of negotiation. For example, if an agent repeatedly offer sushi as "food for a party," it is reasonable to infer that it prefers sushi.

More specifically, to estimate the opponent's utility function, we first create a frequency map of the values in each issue based on the opponent's offers. Fix an issue k, and let $j \in \{1, \ldots, J_k\}$ denote its values. We write h_k to denote the frequency map for issue k, where $h_k(j)$ is the number of times value j has been proposed by the opponent so far. We also write $j_{\text{most}} \in \arg\max_{j \in \{1,\ldots,J_k\}} h_k(j)$ be an issue with the highest frequency. Finally, Z denotes the sum of all frequencies except for that of j_{most}: i.e., $Z = \sum_{j \neq j_{\text{most}}} h_k(j)$.

We now take as our estimate of f'_k proportional frequencies, as follows:

$$\hat{f}'_k(j) = \begin{cases} 1 & \text{if } j = j_{\text{most}} \\ 0 & \text{else if } Z = 0 \\ \frac{h_k(j)}{Z} & \text{otherwise} \end{cases} . \quad (7)$$

Figure 2 shows two examples of calculating $\hat{f}'_k$ from a frequency map. The value of $\hat{f}'_1$ for sandwiches is 1, because it is the value with the maximum frequency, while $\hat{f}'_1$ for sushi is $\frac{3}{24}$, because the frequency of sushi is 3 and the sum of frequencies of everything but sandwiches is 24.

Similar to how our agent sets the weights θ_k for its own user, our agent also sets the weights for the opponent using Equation 1. Thus, it remains only to specify the way in which we determine the order of importance of the issues to the opponent.

We ascribe greater importance to an issue for the opponent when we have greater certainty over their preferences for that issue. Indeed, our importance ordering heuristic is related to entropy. Formally, we define the importance of issue k to the opponent to be:

$$\text{imp}_k = \frac{\sum_{i=1}^{J_k} \hat{f}'_k(i)\, h_k(i)}{\sum_{i=1}^{J_k} h_k(i)} = \frac{\sum_{i=1}^{J_k} \hat{f}'_k(i)\, h_k(i)}{Z + h_k(j_{\text{most}})} . \quad (8)$$

In Figure 2, we also calculate the imp_k values for food and drinks from our sample frequency map. Note that normalization factor is the same for both issues (46), but the agent is more certain about the opponent's preferences over issue 1, as the values in the frequency map are more spread out. Using this formula, issues for which the agent is more certain about the opponent's preferences result in larger imp_k values.

We have thus explained how our agent estimates the values $\hat{f}'_k(j)$ of each value j of each issue k, and how our agent estimates the weights θ_k for each issue. Combining this information linearly, *agentKT* constructs an estimate $\hat{f}$ of the opponent's utility function.

4 ELICITATION METHOD

The preference elicitation problem, i.e., the search for queries to place that yield the greatest utility gain, has been a subject of study for decades (e.g., [4]). The variant of this problem of interest here, however, is one in which elicitation is coupled with negotiation (e.g., [3] and [7]). In particular, potential utility gains must be evaluated with respect to negotiation outcomes, which depend in turn on negotiation strategies. Put slightly differently, given a solution to the problem of inferring the user's utility function (Section 2), the elicitation problem asks how to best extend the partial ranking so that the agent can infer whatever is most relevant about the user's utility function to make the negotiation a success.

Recall that, in ANL, agents begin with a partial picture of the user's preferences in the form of a ranking R of a subset of the outcomes of size n. To obtain more information about the user's utility function, agents can query the user at some cost. Each such query sends a single outcome ω and the current ranking R, and receives an updated ranking R' that includes ω, say at position i (of $n+1$): i.e., $R'(\omega) = i$. Placing such a query is equivalent to asking n comparison questions, each one comparing ω to one of the n outcomes in R. These n additional pairs can be used to update the agent's estimate of the user's utility function using, for example, the method described in Section 2.

Because more principled approaches that consider elicitation as a sequential decision problem (e.g., [4]) are too computationally intensive, we instead describe a popular heuristic approach to the preference elicitation problem in the negotiation context (e.g., [3] and [7]), which employs a greedy policy. This policy continually poses a query that maximizes a quantity called *expected value of information (VOI)*, as long as the expected VOI exceeds the elicitation cost. We now define expected VOI.

issue 1: food for a party

values	pizza	sushi	pasta	sandwiches
h_1	0	3	21	22
f'_1	0	$\frac{3}{24}$	$\frac{21}{24}$	1
imp_1	$\frac{0 \cdot 0 + 3 \cdot \frac{3}{24} + 21 \cdot \frac{21}{24} + 22 \cdot 1}{46} \cong 0.89$			

issue 2: drinks for a party

values	tea	water	pop	juice
h_2	4	9	11	22
f'_2	$\frac{4}{24}$	$\frac{9}{24}$	$\frac{11}{24}$	1
imp_2	$\frac{4 \cdot \frac{4}{24} + 9 \cdot \frac{9}{24} + 11 \cdot \frac{11}{24} + 22 \cdot 1}{46} \cong 0.68$			

Figure 2: Example $\hat{f}'_k$ and $\mathrm{imp_k}$ derivations of issues with different frequencies (h_k's).

Assume an opponent with a fixed negotiation strategy that is known to the agent.[1] Assume further that the agent can compute a best response to this strategy. Note that this best response varies with the user's utility function, and that the agent's estimate of the user's utility function varies with its information about the user, and hence, the user's responses to the agent's queries.

To determine the (myopic[2]) value of a query, the agent need only compute the difference between two competing quantities: the value of its best response to the opponent's negotiation strategy, given its current estimate, $\hat{f}$, of the user's utility function, say $V^*(\hat{f})$; and the value of its best response to the opponent's negotiation strategy, given an updated estimate, $\hat{f}_i$, of the user's utility function based on the response to the query (e.g., position i), say $V^*(\hat{f}_i)$. The quantity $V^*(\hat{f}_i) - V^*(\hat{f})$ is called the *value of information (VOI)*.

One difficulty with computing VOI is that the response to a query is not known until after the agent places the query. What may be known (or possible to estimate), however, is a probability distribution p over where in the partial ranking the response might fall. As each possible position i gives rise to an alternative utility function $\hat{f}_i$, we can thus compute the expected value $V^*(p) = \sum_{i=1}^{n} p(R(\omega) = i)V^*(\hat{f}_i)$, and likewise, the *expected VOI*, $V^*(p) - V^*(\hat{f})$, where $p(R(\omega) = i)$ denotes the probability outcome ω lies in position i.

Myopic Problem Statement. Compute an optimal query, by finding one that maximizes (myopic) expected VOI.[3] □

AgentKT does not explicitly employ a greedy preference elicitation strategy, or compute VOI *per se*. One reason for this is, doing so would require that the agent learn the opponent's negotiation strategy, but it is difficult, if not impossible, to accomplish this feat in ANL. Nonetheless, our agent uses a very basic heuristic elicitation method that can be interpreted as roughly approximating such a greedy policy: it places queries whose information content we expect to exceed their cost, as determined anecdotally by our personal experience building negotiation agents for ANL.

In particular, we surmised that a certain $K + 3$ queries are likely to be high in information content.[4] The first K of these queries are used to determine the requisite per-issue weights $\hat{\theta}_k$ (i.e., the estimated importance of the K issues to the user; see Section 2), a necessary input to our regression module. After issuing these K queries, the per-issue weights and the initial partial ranking are fed to the regression module, to infer an estimate of the user's utility function. Then, three additional queries are placed, after each of which the same per-issue weights and the incrementally expanded rankings are again fed to the regression module, to improve the estimate of the user's utility function. After this fourth call to the regression module, the resulting utility function estimate is used throughout the remainder of the negotiation.

It remains to specify these $K+3$ queries. *AgentKT* determines the first K queries as follows: for all $k \in \{1 \ldots, K\}$, the query is almost ω_{best}, except that the value of the kth issue in ω_{best}, i.e., $\pi_k(\omega_{\text{best}})$, is replaced with $\pi_k(\omega_{\text{worst}})$. In the example shown in Figure 3, this amounts to querying the following outcomes:

$$\omega_{\text{change\_location}}, \omega_{\text{change\_food}}, \omega_{\text{change\_dessert}}, \omega_{\text{change\_drinks}}, \omega_{\text{change\_music}} \ . \quad (9)$$

The user's replies to these queries reveal to the agent the relative importance of the K issues. For example, if the user's preference is $\omega_{\text{change\_music}} < \omega_{\text{change\_location}} < \omega_{\text{change\_dessert}} < \omega_{\text{change\_drinks}} < \omega_{\text{change\_food}}$, it indicates that changing the value of the issue "music" from "jazz" to "classical," as in $\omega_{\text{change\_music}}$, results in the greatest decrease in utility, and thus the issue music is most important. Hence, in this example, the order of importance of the issues is: food < drinks < dessert < location < music. This order of importance of the issues of our agent is used to derive the per-issue weights θ_k.

After issuing these K queries, *agentKT* places queries on three additional outcomes. These outcomes are the first three outcomes not yet in the partial ranking that *agentKT* chooses to offer through the offering strategy described in Section 5. As the elicitation thread is separate from the negotiation thread, it is not guaranteed, but it is likely, that these three outcomes are determined by the offering strategy that is in play during the discussion phase. According to *agentKT*'s current estimate of the user's utility function, these outcomes are expected to yield high utility. The agent places these queries to confirm this suspicion.

[1] The opponent's negotiation strategy is not usually known to the agent, but it may be possible to learn this strategy via repeated negotiations. However, this learning capability is not available in the current rendition of ANL.

[2] This approach is heuristic, because it only computes myopic query values. It does not attempt to evaluate query policies (i.e., sequences of queries). These myopic values are typically combined with a greedy query policy.

[3] We refer the reader to [3] and [7] for a more formal treatment of value of information.

[4] The number three is arbitrary (i.e., it is not experimentally grounded); we did not have time to optimize this hyperparameter before the 2020 competition.

	location	food	dessert	drinks	music
ω_{best}	park	sandwiches	ice cream	juice	jazz
ω_{worst}	restaurant	pizza	popcorn	tea	classical
$\omega_{\text{change\_location}}$	restaurant	sandwiches	ice cream	juice	jazz
$\omega_{\text{change\_food}}$	park	pizza	ice cream	juice	jazz
$\omega_{\text{change\_dessert}}$	park	sandwiches	popcorn	juice	jazz
$\omega_{\text{change\_drinks}}$	park	sandwiches	ice cream	tea	jazz
$\omega_{\text{change\_music}}$	park	sandwiches	ice cream	juice	classical

Figure 3: Example of K possible initial queries (i.e., the first K of our agent's $K+3$ queries).

5 NEGOTIATION STRATEGY

Undertaking a game-theoretic analysis of the negotiation problem posed by ANL does not appear to be tractable. Were we able to carry out such an analysis to arrive at a game-theoretic equilibrium, that equilibrium would provide a prediction about the opponent's behavior. There are other ways to make such predictions, such as learning over the course of repeated negotiations, but repeated negotiation is not currently a feature of ANL.

Barring much opportunity for immediate success by pursuing either of these two more-principled alternatives, our approach was to do our best to develop a robust heuristic strategy that performs well empirically: i.e., that negotiates well with the agents that are typical of ANL, most of which are heuristic time-based strategies [5]. *AgentKT* thus also employs a time-based strategy, which incorporates its estimates of *both* the user's and the opponent's utility functions.

AgentKT's time-based strategy makes its decisions as a function of the time t, where t = (Current Round)/(Total Number of Rounds). We call the first T rounds the *initial phase*, round $T+1$ to $t \leq 0.5$ the *discussion phase*, $0.5 < t \leq 0.95$ the *consideration phase*, $0.95 < t < 1$ the *joint preference phase*, and $t = 1$ the *final round*. A summary of these phases and the agent's corresponding strategies is shown in Figure 4.

Initial phase. Our agent is in the process of eliciting information to estimate its user's utility function. It always offers ω_{best}, and does not accept any offers from the opponent.

Discussion phase. Our agent obtains information about the tendencies in the opponent's offer and lets the opponent know about its own preferences. It offers an outcome uniformly at random from a time-dependent set $\mathcal{X} = \{\omega \mid \hat{f}(\omega) > 0.95 - 0.5 \cdot t^2, \omega \in \Omega\}$, and does not accept any offers from the opponent.

Consideration phase. Our agent uses its estimates of its user and the opponent's utility functions when offering outcomes, and accepts offers based on the time-dependent threshold developed in agent *Atlas3* [12] from ANAC 2015, which we describe here for completeness. We could not find this strategy explained in prose anywhere, so this description derives from our understanding of the *Atlas3* code.

Let $u_{\text{joint}}(t)$ be the expected joint utility of both the agent and its oppponent at time t, with additional time-dependent weight on its own utility as follows: $u_{\text{joint}}(t) = (1.8 - 0.3 \cdot t^2)\hat{f}(\omega) + \hat{f}'(\omega)$. As t increases from 0 to 1, the weight on its own utility decreases from 1.8 to 1.5, while the weight on the opponent's utility remains 1. This weighting ascribes greater weight to the agent's own utility at all times $t \in [0, 1]$, while allowing the relative weight of the opponent's utility to increase over time.

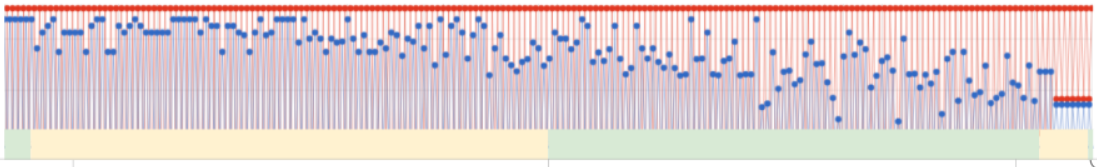
Let $\omega_{\text{maxJoint}}(t)$ be an outcome with the maximum joint utility u_{joint}, at time t. That is, $\omega_{\text{maxJoint}}(t) \in \arg\max_{\omega \in \mathcal{X}} u_{\text{joint}}(t)$. Our agent offers an outcome uniformly at random from the set $\mathcal{Y} = \{\omega \mid \hat{f}(\omega) \geq \hat{f}(\omega_{\text{maxJoint}}(t)), \omega \in \mathcal{X}\}$.

Figure 5 shows an example of deriving the set $\mathcal{Y}$ from $\hat{f}$s and $\hat{f}'$s at times $t = 0.7$ and $t = 0.9$. We can see that outcome ω_B has the maximum non-weighted joint utility $\hat{f} + \hat{f}'$. However, at $t = 0.7$, $\omega_{\text{maxJoint}} = \omega_D$, since $\hat{f}(\omega_D) = 0.98$ and large weight is ascribed to $\hat{f}$. At $t = 0.9$ though, $\omega_{\text{maxJoint}} = \omega_B$, because the weight on $\hat{f}$ decreases as t increases, so relative to before, more weight is put on $\hat{f}'$. In sum, the size of the set $\mathcal{Y}$ increases over time, and our agent likewise places more generous offers.

The acceptance strategy of *Atlas3* is a time-dependent thresholding strategy. In a thresholding strategy, an agent accepts any offer above a certain threshold b, which usually derives from its user's utility function and time. For *Atlas3*, this threshold is a linear function of time, namely $b = 1 - t \cdot (1 - \mathbb{E}[u_{\text{final}}])$, where $\mathbb{E}[u_{\text{final}}]$ is the user's expected final utility. Equivalently, $b = \mathbb{E}[u_{\text{final}}] + (1 - \mathbb{E}[u_{\text{final}}])(1 - t)$. Based on this equation, b is initialized to 1 and then decreases to $\mathbb{E}[\hat{u}_{\text{final}}]$ as t increases from 0 to 1.

To calculate $\mathbb{E}[\hat{u}_{\text{final}}]$, an estimate of $\mathbb{E}[u_{\text{final}}]$, *Atlas3* first constructs a prediction $q \in [0, 1]$ of the likelihood that the opponent is hardliner (where $q = 0$ indicates low likelihood and $q = 1$ indicates high likelihood). This prediction is based on the estimated utility of the opponent's best offer. Specifically, q is highest when the estimated utility of the opponent's best offer is lower than our agent's reservation value, and decreases as the estimated utility of the opponent's best offer increases. *Atlas3* then estimates its expected final utility assuming it concedes in the final round: $\mathbb{E}[\hat{u}_{\text{final}}] = qu(\text{CH}) + (1 - q)u(\text{CC})$. Here, $u(\text{HC})$ is *Atlas3*'s final utility when it is a hardliner and the opponent is a conceder, while $u(\text{CH})$ is defined symmetrically. The notation $u(\text{HH})$ and $u(\text{CC})$ are likewise defined, and

$$q = \frac{1}{1 + \frac{u(\text{CH}) - u(\text{HH})}{u(\text{HC}) - u(\text{CC})}} \quad . \tag{10}$$



	initial phase	discussion phase	consideration phase	joint phase	final round
Bidding Strategy	ω_{best}	random from $\mathcal{S} = \{\omega \mid \hat{f}(\omega) > 0.95 - 0.5 \cdot time^2, \omega \in \Omega\}$	random from $\mathcal{T} = \{\omega \mid \hat{f}(\omega) > \max(0.95 - 0.5 \cdot time^2, \omega_{maxjoint}), \omega \in \mathcal{S}\}$	$\omega_{maxjoint}$	
Acceptance Strategy	doesn't accept any offers		over $\mathbb{E}[u_f] + (1 - \mathbb{E}[u_f])(1 - time)$		over reserve

Figure 4: Summary of phases and corresponding strategies. The blue dots represent *agentKT*'s utility of its offers during a negotiation. The red dots represent the utility of the opponent, for the opponent's offers, except during the joint phase, where the opponent's utility for *agentKT*'s offers is also shown. In this negotiation, the opponent is a hardliner.

t = 0.7					
outcomes	ω_A	ω_B	ω_C	ω_D	ω_E
$\hat{f}$	0.58	0.71	0.86	0.98	0.99
$\hat{f}'$	0.86	0.90	0.64	0.46	0.31
$\hat{f}+\hat{f}'$	1.44	1.61	1.50	1.44	1.30
X	$= \{\omega \mid \hat{f}(\omega) > 0.95 - 0.5 \cdot t^2 = 0.705, \omega \in \Omega\} = \{\omega_B, \omega_C, \omega_D, \omega_E\}$				
u_{joint}	1.82	2.07	2.06	2.08	1.95
$\omega_{maxjoint}$	$= \text{argmax}_{\omega \in X} u_{joint} = \omega_D$				
Y	$= \{\omega \mid \hat{f}(\omega) \geq \hat{f}(\omega_{maxjoint}) = \omega_D, \omega \in X\} = \{\omega_D, \omega_E\}$				

t = 0.9					
outcomes	ω_A	ω_B	ω_C	ω_D	ω_E
$\hat{f}$	0.58	0.71	0.86	0.98	0.99
$\hat{f}'$	0.86	0.90	0.64	0.46	0.31
$\hat{f}+\hat{f}'$	1.44	1.61	1.50	1.44	1.30
X	$= \{\omega \mid \hat{f}(\omega) > 0.95 - 0.5 \cdot t^2 = 0.545, \omega \in \Omega\} = \{\omega_A, \omega_B, \omega_C, \omega_D, \omega_E\}$				
u_{joint}	1.76	2.01	1.98	1.99	1.85
$\omega_{maxjoint}$	$= \text{argmax}_{\omega \in X} u_{joint} = \omega_B$				
Y	$= \{\omega \mid \hat{f}(\omega) \geq \hat{f}(\omega_{maxjoint}) = \omega_B, \omega \in X\} = \{\omega_B, \omega_C, \omega_D, \omega_E\}$				

Figure 5: Sample ω_{maxJoint} and $\mathcal{Y}$ derivations.

Let ω_{reserve} be the reservation outcome and let $\omega_{\text{bestOffered}}$ be the outcome with the highest estimated utility so far. *Atlas3* instantiates the variables in Equation 10 as follows:

- $u(\text{CH})$ is *agentKT*'s final utility when it is a conceder and its opponent is a hardliner. *Atlas3* sets CH equal to the maximum of $\hat{f}(\omega_{\text{reserve}})$ and $\hat{f}(\omega_{\text{bestOffered}})$.
- $u(\text{HH})$ is *agentKT*'s final utility when both agents are hardliners, which is the estimated reservation value $\hat{f}(\omega_{\text{reserve}})$, since any such negotiation would end without either party accepting an offer.
- $u(\text{HC})$ is *agentKT*'s final utility when it is a hardliner and its opponent is a conceder. *Atlas3* takes this value to be 1, which is the maximum possible utility of any outcome.
- $u(\text{CC})$ is *agentKT*'s final utility when both agents are conceders. As either of the two agents can concede first, with equal probability, *Atlas3* takes CC to be equal to $\frac{1}{2} \cdot \text{CH} + \frac{1}{2} \cdot \text{HC}$.

Plugging back in to the formula for q (Equation 10) yields:

$$q = \frac{1}{1 + 2\left(\frac{\max(\hat{f}(\omega_{\text{reserve}}), \hat{f}(\omega_{\text{bestOffered}})) - \hat{f}(\omega_{\text{reserve}})}{1 - \max(\hat{f}(\omega_{\text{reserve}}), \hat{f}(\omega_{\text{bestOffered}}))}\right)}$$

The fractional term in the denominator is 0 when $\hat{f}(\omega_{\text{BestOffered}}) \leq \hat{f}(\omega_{\text{reserve}})$, and is otherwise unbounded as $\hat{f}(\omega_{\text{BestOffered}})$ approaches 1. Therefore, q is 1 when $\hat{f}(\omega_{\text{BestOffered}}) \leq \hat{f}(\omega_{\text{reserve}})$, and tends to 0 as $\hat{f}(\omega_{\text{BestOffered}})$ increases. More specifically, when $\hat{f}(\omega_{\text{BestOffered}}) > \hat{f}(\omega_{\text{reserve}})$, then q is a function of where $\hat{f}(\omega_{\text{BestOffered}})$ lies relative to the minimum utility $\hat{f}(\omega_{\text{reserve}})$ and the maximum utility 1. As $\hat{f}(\omega_{\text{BestOffered}})$ approaches its minimum, q increases, as the opponent is behaving more like a hardliner. But when $\hat{f}(\omega_{\text{BestOffered}})$ approaches its maximum, q decreases, as the opponent is behaving more like a conceder. Figure 6 shows two sample q derivations which exemplify this behavior.

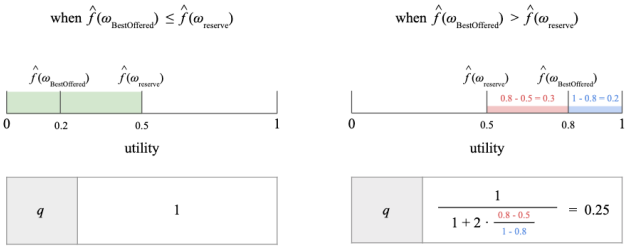


Figure 6: Two sample q derivations, one per case.

Joint preference phase. Our agent offers outcome ω_{maxJoint}, and accepts under the same conditions as in the *consideration phase*.

Final round. Our agent offers outcome ω_{maxJoint}, and accepts any offer that has a greater utility than its reservation value, which is estimated by evaluating $\hat{f}$ at the reservation outcome, which the agent is given.

Rank	Agent Name	Utility	Penalty	Score
1	AhBuNe Agent	0.6623472	0.0069614	0.6553858
2	Hamming agent	0.64844519	0	0.64844519
3	Shine agent	0.65906589	0.01867582	0.64039007
4	AgentKT	0.66398437	0.03039321	0.63359116
5	ANGEL party	0.60963705	0	0.60963705

Table 1: Overall ranking of the finalists (the five top-performing agents) in the ANAC 2020 Automated Negotiation League.

6 EVALUATION

Having described *agentKT*'s strategy, we now present two experiments that demonstrate its behavior. As in Figure 4, in the plots in this section, the blue dots represent *agentKT*'s utility, while the red dots represent the opponent's utility.

vs. agentKT. In this run (Figure 7), we observe that when running against itself, *agentKT* performs very well, achieving a high individual utility (0.9501 and 0.9088) as well as a high joint utility (1.8589). This is likely due to the two agents modelling their opponent's utility function in the same way the opponent models its own utility function.

vs. Hardliner In this run (Figure 8), we can see how *agentKT* gradually changes its bidding strategy. Although no agreement is achieved in this negotiation, because the hardliner opponent only offers and accepts the maximum utility outcome, *agentKT* nonetheless eventually finds an outcome that yields high utility (~0.78) for both parties.

7 COMPETITION PERFORMANCE

ANAC 2020 was held as part of the 29th International Joint Conference on Artificial Intelligence. Table 1 lists the ranking of all finalists in the Automated Negotiation League. *AgentKT* performed well, earning the 4th highest overall score, which is calculated by subtracting the agent's average elicitation penalty from its average utility. In fact, *AgentKT* earned the highest average utility value, but at the same time, it incurred substantially higher elicitation costs than any of the other agents. This large expense was a result of its elicitation method, which placed a fixed number of queries, regardless of the initial size of the partial ranking. The fact that two of the five finalists incurred zero elicitation costs suggests that elicitation was not actually essential to performing well in the competition. On the contrary, it appears that the initial partial rankings already contained sufficient information about the user's utility function. We would not expect this to be the case in general; on the other hand, an elicitation method should be flexible enough to recognize when this is the case and curtail its questioning accordingly.

8 CONCLUSION

The Automated Negotiating Agents Competition (ANAC), like most competitions of its ilk, is intended to foster research in a domain, by creating a common problem for researchers to address. The problem, in the case of ANAC's Automated Negotiation League, is to design an agent strategy for a simulated negotiation setting. The negotiation task alone is notoriously difficult, even without the added complications posed by mixing in the preference elicitation task. Consequently, a principled approach to deriving a strategy, such as solving for an equilibrium of this game, is not likely to yield fruit in the short term. Alternatively, it is possible to break this complex problem down into various constituent parts, address each of those in a principled way, and piece them together to arrive at a semi-principled overall agent strategy.

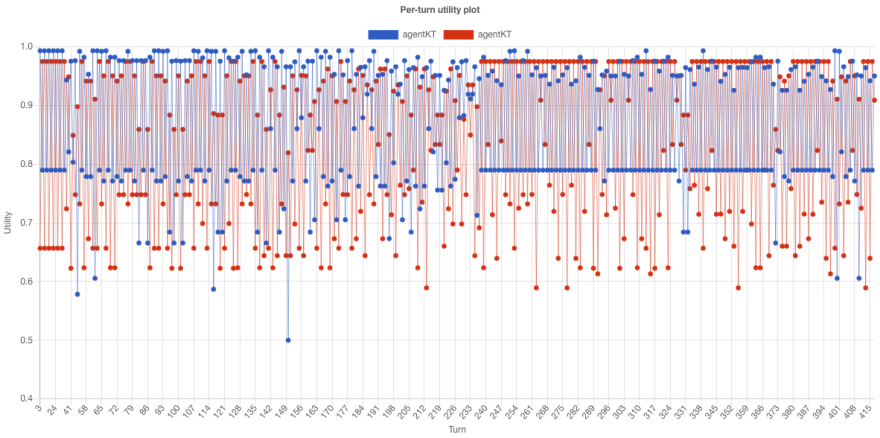
The goal of this paper is to describe the constituent parts of the problem posed by ANL 2020, and to propose baseline heuristics to solve each of these parts. To this end, we described *agentKT*, an automated negotiation agent that was a finalist in this league. The agent comprises four basic modules. The user's utility function is learned by converting a partial ranking of outcomes into comparison data, which then defines a regression problem that we solve using linear programming. A model of the opponent's utility function is built by using the opponent's offers to construct a measure of uncertainty like entropy. Our elicitation method places a fixed number of carefully constructed queries, each of which generates comparisons across multiple outcomes, to be fed to our regressor. Finally, our negotiation strategy is time-dependent.

As we provide only baseline heuristic solutions for these core subproblems, there are many directions for future work. For example, we are already testing our linear regression method against alternative linear and non-linear regressors that do not rely on a heuristic to set the values of the θ_k's. Preliminary experiments show that a linear regression that is not constrained by heuristically-determined settings of the θ_k's—i.e., one that instead directly incorporates these weights into the per-value estimates $\hat{f}_k$'s—performs more accurately than the linear regression we used in *agentKT* in 2020. Likewise, a non-linear regression that simultaneously optimizes both f_k's and θ_k's also outperforms our method, but only at a substantially increased computational cost. In future work, we plan to generalize the inference problem as one of learning a probability distribution over, instead of a point estimate of, a utility function. Further, we plan to extend our solution techniques to learn the opponent's utility function as well as the user's.

Future work on the preference elicitation problem includes implementing a variation of the Fast VOI Elicitation algorithm [11] to decide which outcomes to elicit. Our current method elicits a *fixed* number of outcomes, namely initial K outcomes and an additional three outcomes which the agent estimates have high expected utility. In contrast, the Fast VOI Elicitation algorithm computes the expected VOI of all queries, and then elicits an outcome with maximal expected VOI, if its expected VOI is greater than the cost of elicitation. We hope and expect that these improvements will increase the negotiating power, and ultimately, the utility of *agentKT* in future renditions of ANL.

Notably absent from this list of core subproblems is modelling the opponent's negotiation strategy [1]; only the opponent's utility function is modelled. Modelling the opponent's behavior is difficult in the ANL; as negotiations are not repeated, the only opportunity to learn about their behavior is during one, short negotiation. We look forward to addressing this problem in future versions of ANL that incorporate repeated negotiations (e.g., ANL 2021).

While there are many principled ways to envision improving upon our heuristics, the primary goal of this work was to identify



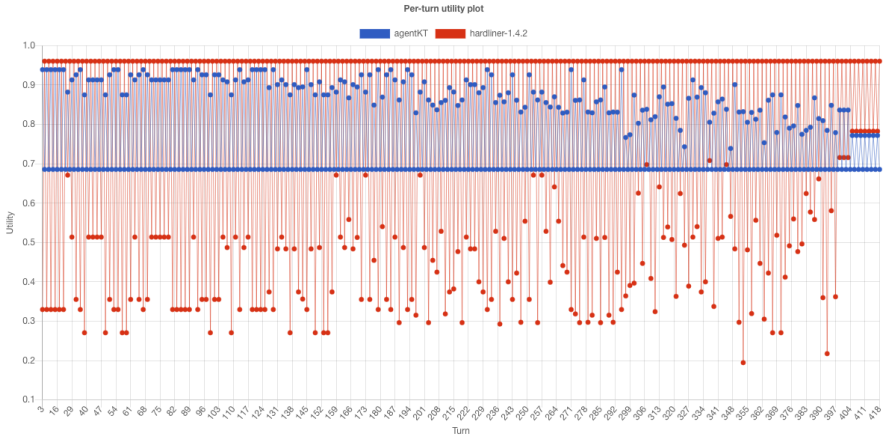
Agreement

Round: 191

Issue	Agreed Value
Invitations	
Music	
Drinks	
Cleanup	
Food	
Location	

Party	Profile	Utility
agentKT	party3	0.9501
agentKT	party4	0.9088

Figure 7: agentKT vs. agentKT. Agreement is reached in round 191.



No Agreement

Round: 200

Figure 8: agentKT vs. Hardliner. No agreement is reached.

the core subproblems inherent in automated negotation, and to provide heuristic solutions to these subproblems that can serve as baselines in future runs of the competition. Indeed, the ANL organizers are planning to incorporate our code modules into an upcoming release of their platform for exactly this reason—so that future developers can benchmark their solutions against a community standard. This paper is intended to accompany those code modules, by providing an English description of these baselines. We hope this paper will also serve to encourage more AAMAS researchers to participate in future renditions of ANAC by easing the barrier to entry, and by clarifying what progress in automated negotiation means, as something akin to increasing accuracy on a supervised learning task.

REFERENCES

[1] 2016. Learning about the opponent in automated bilateral negotiation: a comprehensive survey of opponent modeling techniques. *Autonomous Agents and Multi-Agent Systems* 30, 5 (1 Sept. 2016), 849–898. https://doi.org/10.1007/s10458-015-9309-1

[2] Reyhan Aydoğan, David Festen, Koen V Hindriks, and Catholijn M Jonker. 2017. Alternating offers protocols for multilateral negotiation. In *Modern Approaches to Agent-based Complex Automated Negotiation*. Springer, 153–167.

[3] Tim Baarslag and Michael Kaisers. 2017. 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*. International Foundation for Autonomous Agents and Multiagent Systems, 391–400.

[4] Craig Boutilier. 2002. A POMDP formulation of preference elicitation problems. In *AAAI/IAAI*. 239–246.

[5] Peyman Faratin, Carles Sierra, and Nick R Jennings. 1998. Negotiation decision functions for autonomous agents. *Robotics and Autonomous Systems* 24, 3-4 (1998), 159–182.

[6] Cuihong Li, Joseph Giampapa, and Katia Sycara. 2006. Bilateral negotiation decisions with uncertain dynamic outside options. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 36, 1 (2006), 31–44.

[7] Yasser Mohammad and Shinji Nakadai. 2019. Optimal Value of Information Based Elicitation During Negotiation. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems* (Montreal QC, Canada) *(AAMAS '19)*. International Foundation for Autonomous Agents and Multiagent Systems, 242–250.

[8] John F Nash Jr. 1950. The bargaining problem. *Econometrica: Journal of the Econometric Society* (1950), 155–162.

[9] Michael J. D. Powell. 1994. A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation. *Advances in Optimization and Numerical Analysis* (1994), 51–67. https://doi.org/10.1007/978-94-015-8330-5_4

[10] Ariel Rubinstein. 1982. Perfect equilibrium in a bargaining model. *Econometrica: Journal of the Econometric Society* (1982), 97–109.

[11] Yasser Mohammad Shinji Nakadai. 2018. FastVOI: Efficient Utility Elicitation During Negotiations. *PRIMA 2018: Principles and Practice of Multi-Agnet Systems* 11224 (2018), 560–567. https://doi.org/10.1007/978-3-030-03098-8_42

[12] Akiyuki Mori Takayuki Ito. 2016. *Atlas3: Anegotiating Agent Based on Expecting Lower Limit of Concession Function*. 169–173.