Negotiating Concurrently with Unknown Opponents in Complex, Real-Time Domains

Colin R. Williams and **Valentin Robu** and **Enrico H. Gerding** and **Nicholas R. Jennings**[1]

Abstract. We propose a novel strategy to enable autonomous agents to negotiate concurrently with multiple, unknown opponents in real-time, over complex multi-issue domains. We formalise our strategy as an optimisation problem, in which decisions are based on probabilistic information about the opponents' strategies acquired during negotiation. In doing so, we develop the first principled approach that enables the coordination of multiple, concurrent negotiation threads for practical negotiation settings. Furthermore, we validate our strategy using the agents and domains developed for the International Automated Negotiating Agents Competition (ANAC), and we benchmark our strategy against the state-of-the-art. We find that our approach significantly outperforms existing approaches, and this difference improves even further as the number of available negotiation opponents and the complexity of the negotiation domain increases.

1 INTRODUCTION

Recent years have seen an increasing interest in developing automated bargaining strategies that allow autonomous agents to negotiate, on behalf of their owners, in complex, realistic environments. In particular, these environments are characterised by having (1) concurrent negotiations with multiple opponents, who are in turn negotiating with other opponents; (2) negotiations involving multiple issues; (3) continuous time, where negotiation proceeds in real time as opposed to fixed rounds; and (4) agents with no prior knowledge about their opponents. In this paper we deal with all these issues simultaneously and consider, for the first time, a principled negotiation strategy for such complex environments.

In more detail, there is a growing body of work that considers strategies for multi-issue negotiations against unknown opponents, by employing a variety of machine learning and other AI techniques [6, 7, 9]. However, a significant shortcoming of these works is that they consider only bilateral, one-to-one negotiations. In practice, agents are often required to negotiate concurrently with multiple opponents. This is challenging since the strategy of one opponent may depend on what is happening in other negotiation threads. Furthermore, when negotiations are *many-to-many*, i.e. where all agents have multiple negotiation opportunities, an agent may suddenly leave the negotiations if they have reached an agreement with another opponent. As a result, delays by the participants put them at risk of failing to reach an agreement with a particular opponent. Finally, if the protocol allows for *decommitment*, even if an offer is accepted, an agent may continue to negotiate with other opponents in the hope of reaching an even better deal.

In the existing literature, there have been several papers that investigate the problem of one-to-many (where the opponent is not negotiating with any other agent) and many-to-many negotiations. Aknine et al. [1] propose an extension of the well-known contract net protocol which enables many-to-many negotiations to terminate quickly and and is tolerant to crash failures. However, they do not consider the negotiation strategies, but only the properties of the protocol (e.g., whether any deadlocks occur). Negotiation strategies are considered by Giampapa et al. [6] who extend some of the concession heuristics proposed for bilateral negotiation to account for dynamic, outside options. However, their work does not consider the negotiation opponents explicitly, only through exogenous probability distributions.

The most related prior works are An et al. [2], An et al. [3] and Nguyen and Jennings [7]. An et al. [3] is the first work to consider continuous-time, one-to-many negotiations. They propose a heuristic negotiation strategy and focus on finding good waiting-time strategies before making a proposal. However, their work only considers single-issue negotiations and is based on combining a number of ad-hoc heuristics containing a large number of parameters. In contrast, we consider a more principled approach and environments with multiple issues. The latter increases the uncertainty about the opponent, and so requires exploring the outcome space. This considerably reduces the benefit of waiting strategies. In a different work, An et al. [2] derive Nash equilibrium strategies for agents participating in one-to-many and many-to-many negotiations. However, their work assumes single-issue negotiations, discrete time, and focuses primarily on complete information settings. Thus their work is not appropriate for practical negotiation settings with multi-issue negotiations, continuous time, and with no prior knowledge about the opponents. In contrast, Nguyen and Jennings [7] propose a practical negotiation heuristic for one-to-many negotiations with uncertainty about the opponents. However, their approach only considers discrete time, makes strong assumptions about the opponents, and requires considerable prior knowledge about these opponents. In particular, they assume that there is a small number of different opponent *types*, all using a simple time-based concession strategy. Furthermore, they assume that the probabilities of each type are known, as well as the payoff that will be obtained when negotiating against each type. In this work, we use their approach as a benchmark, after considerably extending it to handle multi-issue domains and real time. Despite these adjustments, and relying on prior knowledge, we show that our novel strategy developed considerably outperforms the approach of [7].

Against this background, our aim is to develop a practical negotiation heuristic using a principled approach for negotiating concurrently against a range of unknown opponents. Specifically, we assume that both the utility functions and the behaviours of the opponents are unknown. This work is the first to study many-to-many negotiations against unknown opponents in large, multi-issue domains. In more detail, our contributions to the state-of-the-art are as follows:

- We propose a novel negotiation strategy that allows the coordination of multiple, concurrent negotiation threads against unknown opponents, in complex multi-issue domains, and using a princi-

[1] University of Southampton, United Kingdom, email: {crw104, vr2, eg, nrj}@ecs.soton.ac.uk

pled approach. In particular, we formulate the strategy as an optimisation problem where, in real time, the optimal utility level is computed at which (multi-issue) offers are generated. The decisions are made based on probabilistic information about the opponents, which is updated over time after observing offers received.
- We consider two variants of our strategy: one in which potentially different utility levels are adopted in each negotiation thread, and one in which the utility level is the same across all threads (and so the optimisation problem is reduced to a single dimension).
- Using simulations, we extensively validate our approach against opponents from the recent international negotiating agents competition (ANAC), which we adapt to our setting. Moreover, we benchmark the performance of the two variants of our strategy against a simple strategy that randomly sets its target utility level, as well as the state-of-the-art [7]. In particular, we show that we outperform the random strategy by 20%-48% and [7] by 13%-26%, depending on the number of concurrent negotiations.

The remainder of this paper proceeds as follows. Section 2 outlines the negotiation protocol used, while Section 3 describes the novel strategy that we designed for this setting. We evaluate the strategy in Section 4 and finally, we conclude in Section 5.

2 CONCURRENT NEGOTIATION PROTOCOL

Our many-to-many negotiation protocol is similar to the ones described in [2, 7]. Furthermore, as in [7] we allow for *decommitment*, subject to a penalty, to allow for more flexibility and a fair comparison with the benchmark strategy.

In more detail, negotiation takes place in multiple, concurrent threads, between pairs of agents. In each of these threads, the agents use an alternating offers protocol, in which the possible actions are OFFER, ACCEPT, CONFIRM, END and DECOMMIT. The negotiation begins with the agents exchanging OFFER messages. Each offer, o, represents a complete package and specifies the values for all negotiable issues (e.g. price, delivery, penalties, quality of service). Formally, $o = (v_1, v_2, \ldots, v_n)$, where v_i is the value for issue i. Sending an OFFER message in response to an OFFER from the opponent constitutes a *counter*offer and implicitly a rejection of the previous offer. If an agent is satisfied with the most recent OFFER it received, it can send an ACCEPT message in order to indicate that it wishes to form an agreement. Following an ACCEPT message being sent in a negotiation thread, no further OFFER messages can be sent. The only messages allowed at this stage are CONFIRM and END. The CONFIRM message is used to indicate that the agent confirms that a binding agreement has been formed. Instead, the END message will abort the negotiation thread.

The reason for including a CONFIRM message is as follows. In the protocol, an agent is allowed to send offers to multiple opponents at once. Therefore, it may find that, while waiting for a response from them, more than one of these offers are accepted. If the ACCEPT messages were to form a binding agreement at this point, the agent may inadvertently reach more than one agreement, and it would need to decommit from all but one of them, thereby incurring decommitment penalties. In this case, the CONFIRM and END messages can be used to select only one of them. Note that an agent could use this strategically by delaying sending the CONFIRM message. However, the agent is expected to confirm the acceptance within a short period of time (at most few seconds, depending on communication delays). Moreover, the opponent who sent the acceptance is still free to abort the agreement without penalty by using the END message. Provided that an agent does not ACCEPT an opponent's offer whilst it is waiting for another agent to CONFIRM an acceptance (or END a negotiation), the agent can avoid reaching multiple agreements.

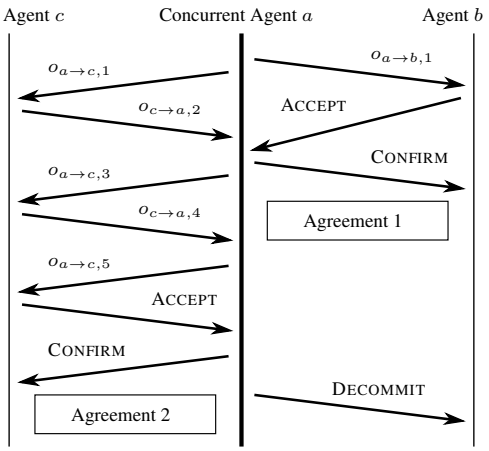


Figure 1. Sequence diagram showing a simplified negotiation trace between three agents, including two agreements and a decommitment.

In a negotiation where there are multiple opponents, it is possible that, after a binding agreement is reached, one of the remaining opponents makes (or accepts) an offer that has a greater utility than that of the existing agreement. In such a situation, it may be beneficial to accept the new offer, and, at the same time, DECOMMIT from the existing agreement. In order to discourage the agents from decommiting unnecessarily, we introduce a decommitment penalty, which is paid by the agent that chooses to decommit from a binding agreement. Without such a penalty, all agreements would essentially become non-binding, leading to a potentially unstable system. Before a CONFIRM message has been sent within a thread, it is possible for either agent to send an END message in order to walk away from the negotiation thread without an agreement, and no penalty is payable.

To encourage the agents to negotiate without delay, there is a deadline which is known to both agents and is in real time, beyond which no agreements can be formed. Furthermore, a discounting factor is used to reduce the value of an agreement according to the time at which it was formed. Formally, the final utility of an agreement of outcome o at time t is given by $U_{\text{disc}}(o,t) = U(o) \cdot (t/t_{\text{max}})^{\delta}$ where δ is the discounting factor, t_{max} is the deadline and $U(o)$ is the undiscounted utility of outcome o, provided that $t \leq t_{\text{max}}$, otherwise the utility is 0.

Figure 1 shows an example negotiation trace with three agents, where agent a negotiates concurrently with agents b and c. After a sends an offer to b, agent b accepts a's offer. Agent a then confirms and an agreement is reached. Agents a and c continue to negotiate, aiming to find an agreement that is better than the existing one (taking into account the decommitment penalty). After a total of five offers have been exchanged, agent c accepts a's offer. Agent a then confirms this agreement, and simultaneously decommits from the worse agreement with agent b. In practice, negotiation traces are likely to be considerably longer.

3 CONCURRENT NEGOTIATION STRATEGY

The strategy that we have developed consists of two key components, which we refer to as the *coordinator* and the *negotiation threads*. This structure is used in order to modularise the information flow within the agent. In more detail, each negotiation thread is responsible for managing the negotiation with a single opponent, using information learnt during the interaction with that opponent, along with information provided by the coordinator. The coordinator is the only

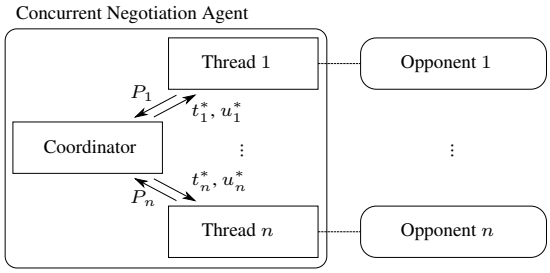


Figure 2. Architecture of the concurrent negotiation agent.

component which deals directly with information provided by all of the negotiation threads, in order to determine the best response across the entire set of opponents. It then uses this information to adjust the behaviour in the individual threads. We discuss the specific details of each of these components in turn, in Sections 3.1 and 3.2, before discussing our approach to handling decommitment in Section 3.3.

3.1 The Negotiation Threads

The strategy of each negotiation thread is an extension of a recently published, principled, adaptive bi-lateral negotiation agent [9]. This agent was designed to be used in a similarly complex environment, but only for negotiations against a single opponent. In more detail, each thread, i, performs Gaussian process regression in order to predict the future concession of its opponent. The prediction is based on the offers received so far by this opponent, and is updated as more offers are received. The Gaussian process enables the prediction to be captured in the form of a probability distribution over the utility, $p_{i,t}(u_i)$ for all future time points, $t \in [t_c, t_{\max}]$ (see [9] for details). The probability distribution is then passed on to the coordinator, which uses it (along with those from other threads) to determine, for each thread i, the best time, t_i^*, at which to reach an agreement, and the best utility, u_i^*, at which the thread should aim to reach the agreement. The way in which the coordinator calculates these values forms a core part of the negotiation strategy, and is discussed in detail in Section 3.2. For now, we will simply take these two values as given.

Given its target time, t_i^*, and target utility, u_i^*, at that time, a negotiation thread needs to: (1) determine the target utility at which to generate offers and to accept incoming offers *right now*[2] and, (2) generate multi-issue offers at the current target utility. Now, to determine the target utility, u_τ, at the current time, t_c, each thread uses polynomial time-dependent concession, where the concession rate is set such that the target utility level reaches u_i^* at time t_i^*.[3] By scaling the utility between 0 and 1, the current target utility is given by:

$$u_\tau(t_c) = 1 - (1 - U_{\min})(t_c/t_{\max})^{1/\beta} \quad (1)$$

where $U_{\min}$ is the minimum utility that the thread will concede to, t_c is the current time, and $t_{\max}$ is the agent's deadline. In our experiments, we set $U_{\min} = 0.5$ since, given the scaling of the utility function between 0 and 1, in the multi-issue negotiations we consider an efficient agreement would give at least as much to the opponent. Therefore, since we are interested in obtaining at least as much utility as our opponent, even though we don't know the opponent's utility function, the agent should never concede below this value. Furthermore, β is set such that (if possible) Equation 1 passes through $[t_i^*, u_i^*]$. Formally,[4]

$$\beta = \log(t_i^*/t_{\max})/\log(1 - u_i^*/1 - U_{\min}) \quad (2)$$

Finally, since we are concerned with multi-issue negotiation, it is necessary to generate a multi-issue offer, o, such that $U(o) \approx u_\tau$. We use the same approach as [9], which is to generate random offers until one is found which has a utility, $U(o) \in [u_\tau - 0.025, u_\tau + 0.025]$. If an offer cannot be found within this range, the range is expanded, until a solution is found. Furthermore, if the target drops below the highest value of the offers made by the opponent, we instead propose the package with that utility that was offered by the opponent. This is since we assume that, for a set of possible offers with utility greater than u_τ, the one which is most likely to be accepted is the one which has previously been offered by the opponent. It may be possible to improve the selection of offers by modelling the preferences of the opponents. However, due to the real-time aspect to the negotiations we consider, we found that using this simple, fast approach to selecting an offer produced very good results.

3.2 The Coordinator

The role of the coordinator is to calculate the best time, t_i^* and utility value, u_i^* at that time, for each thread. To do so, it uses the probability distributions received from the individual threads, which predict future utilities offered by the opponents. In the following, we use $P_{i,t}(u)$ to denote the cumulative probability distribution function, which is the (predicted) probability that the utility of an offer by the opponent will be at least u at time t, and $p_{i,t}(u)$ is the corresponding density function. In addition, recall that the negotiations are many-to-many, and so the opponents may exit the negotiations prematurely if they reach an agreement elsewhere. Since these values cannot be learned during a single negotiation (but can be learned by experimentation from repeated negotiations), we assume that the coordinator has prior knowledge of $P_{c,i}(t, t_c)$, which denotes the probability that opponent i will still be in the negotiation at time $t > t_c$, given that it is in the negotiation at the current time, t_c .

Given this, we formulate the above problem as an optimisation problem, and we consider two related approaches:

1. The first approach is to allow the optimiser to find a different optimal target utility, u_i^*, for each thread. This is a more general approach but, due to the number of variables (one per opponent), finding a solution results in a multi-dimensional optimisation problem and is more computationally intensive. We refer to this approach as the *multiple* u^* strategy.
2. The second approach is to constrain the optimiser to use the same u^* value for all the threads. This is less flexible, as it results in the same behaviour being adopted in all threads. The main benefit is computational, since it results in a single-dimensional optimisation problem. We refer to this approach as the *single* u^* strategy.

To find the optimal strategy, we begin by computing the best time to reach agreement, and then consider the best utility (or utilities), to offer at that time. We do the first part by computing the *expected* utility of an agreement at a given time, and choose the time with the highest expected utility. Although the protocol allows for decommitment, when we compute the expected utility, we simplify

[2] Note that our strategy does not simply delay until t_i^* before making any offers, but uses the intervening time to try and get an even better offer by setting the utility level above u_i^*, and then conceding towards u_i^*.

[3] Note that, importantly, u_i^* and t_i^* are constantly updated by the coordinator, even before the target time is reached, resulting in the concession rate being adjusted as well. Therefore, in practice, the concession over longer periods of time will not be polynomial.

[4] In practice, we also bound $\beta \in [0.01, 2]$ to ensure that the agent never concedes too fast.

the equations by implicitly assuming that we terminate all other threads once an agreement is reached.[5] As a result, a single best time, $t^* \in [t_c, t_{max}]$, is computed for all negotiation threads, as follows:

$$t^* = \text{argmax}_{t \in [t_c, t_{max}]} EU_{rec}(t) \quad (3)$$

where $EU_{rec}(t)$ is the expected utility when reaching an agreement at time t, given by:

$$EU_{rec}(t) = \frac{1}{|A|} \sum_{i \in A} P_{c,i}(t, t_c) \int_0^1 u p_{i,t}(u) \mathrm{d}u \quad (4)$$

where A is the set of *remaining* negotiation threads (i.e. those that have not terminated), and $P_{c,i}(t, t_c)$ is as defined above. Note that the expected utility is computed as the *average* expected utility for each thread. This is because, since we implicitly assume no decommitment, the expected utility assumes we are committed to the first thread that gives us an agreement. Thus, if multiple opponents were to form agreements at roughly the same time, there is an equal probability that any one of those agreements will be formed.

We now look at how u^* is set, given t^*, firstly by introducing the *multiple* u^* approach, then discussing the simpler, *single* u^* one.

3.2.1 Multiple u^*

Given the target agreement point, t^*, we would like to find the optimal utility level for each thread at which to produce offers. To this end, we first specify the expected utility for a given vector of utility levels, one for each (remaining) opponent. We calculate this by assuming that the probability distributions from the various threads are *independently* sampled[6]. Furthermore, as before, we implicitly assume that no decommitment is allowed.

Given this, the expected utility of proposing offers at utility levels $\vec{u}$ at time t can be expressed as:

$$EU_{offer}(\vec{u}, t) = \sum_{A' \in \mathcal{P}(A)} \left(f(\vec{u}, A') \prod_{i \in A'} P_{i,t}(u_i) \prod_{i \in A \setminus A'} (1 - P_{i,t}(u_i)) \right) \quad (5)$$

where A is the set of remaining opponents, u_i is the utility of the offer made to opponent i, $\mathcal{P}(A)$ is the powerset of A, $P_{i,t}(u_i)$ is the probability that opponent i will accept an offer of utility u_i at time t. Note that the right part of the equation denotes the probability of reaching an agreement with *exactly* the agents in the set A' by the time negotiations reach time t. Then, $f(\vec{u}, A')$ is the utility obtained if this occurs. For the same reasons as given above, since we implicitly assume no decommitment, the utility of this event is given by the average of each $u_i, i \in A'$ (since, given that all opponents in A' will accept the offer, the order in which the opponents accept them is equally likely) written formally as $f(\vec{u}, A') = \sum_{i \in A'} \frac{u_i}{|A'|}$.

Given this, we find the set of best values, $\vec{u}^*$, to offer to the opponents by maximising the expected utility. Formally:

$$\vec{u}^* = \text{argmax}_{\vec{u} \in [0,1]^{|A|}} EU_{offer}(\vec{u}, t^*) \quad (6)$$

Since the EU_{offer} function is nonlinear, we use a nonlinear optimisation package (specifically, the *Ipopt* interior point optimizer [8]) to find the solution to Equation 6.

[5] In practice, we do continue to negotiate (as explained in Section 3.3) but this is not captured by the expected utility. In principle, the equations can be extended to include the additional expected utility from decommitment, but this can become computationally intensive to compute, and we leave this for future work.

[6] Note that this is a simplifying assumption and applies to settings where the opponents have widely different strategies and/or preferences. In domains where opponents are similar, these distributions tend to be more correlated.

3.2.2 Single u^*

In the simpler, *single* u^* variant of our coordinator, all negotiation threads are provided with the same value for u^*. As a result, Equation 5 can be simplified to:

$$EU_{offer}(u, t) = \sum_{A' \in P(A), A' \neq \emptyset} \left(u \prod_{i \in A'} P_{i,t}(u) \prod_{i \in A \setminus A'} (1 - P_{i,t}(u)) \right) \quad (7)$$

By further simplification, we get:

$$EU_{offer}(u, t) = u \cdot \left(1 - \prod_{i \in A} (1 - P_{i,t}(u)) \right) \quad (8)$$

In this case, $u^* = \text{argmax}_{u \in [0,1]} EU_{offer}(u, t^*)$.

The benefit of this simplification is that it makes the optimisation problem easier to solve while, depending on the domain, the impact on the optimal expected utility may be limited.

3.3 Handling Decommitment

Although the expected utility does not take into account the possibility of decommitment, in order to benefit from the decommitment option, the agent continues to negotiate with other agents even when an agreement is reached, but will only accept offers which provide a significant improvement even after decommitment penalties are deducted. Since an agent can only ever reach one agreement with each opponent, it is important for the agent to avoid agreements which only provide marginal improvements. This is because any agreement reduces the number of remaining opportunities.

Therefore, once an agreement has been reached, we introduce a minimum utility, u_{min}, for generating offers and accepting an opponent's offer, which is given by the following rule-of-thumb:

$$u_{min} = (u_{existing} + D) * \gamma, \quad (9)$$

where $u_{existing}$ is the utility of the current best agreement, D is the decommitment penalty, and $\gamma > 1$ is a parameter which ensures that the benefit received from the new agreement is sufficiently large. In our agent, we set $\gamma = 1.1$, which means that any new agreement must be worth at least 10% more than the existing one, after paying the penalty. We found this value to work well in practice although, as future work, we hope to set the utility level in a more principled way, by extending the expected utility equations in Section 3.2.

4 EVALUATION

In order to evaluate the performance of our strategy in a realistic and flexible automated negotiation environment, we use the resources provided as part of the GENIUS framework [5]. GENIUS provides a common environment for the development of negotiating agents, and includes a repository of state-of-the-art agents, which we use as negotiation opponents, as we will discuss in Section 4.1. It also provides a range of scenarios, some of which we use in our evaluation (as discussed in Section 4.2). We compare our agent against an existing concurrent negotiation strategy [7], which is less flexible and requires some prior information about the negotiation scenarios and against a simple benchmark (as discussed in Section 4.3). The results of our evaluation are discussed in Section 4.4.

4.1 Evaluation Opponents

To test our strategy in a situation where the behaviour of the opponent is unknown, we require a range of different opponent strategies. To this end, we use the 7 independently developed, state-of-the art

strategies that were finalists in the most recent Automated Negotiating Agent Competition (ANAC2011)[7] [4]. These strategies were all designed for use in complex, real-time negotiations, but against only a single opponent. In order to adapt these agents for the one-to-many protocol, they need to be capable of sending CONFIRM messages. Since the only rational reason not to confirm an acceptance is if the agent has already reached another agreement, adding this functionality to the existing agents is straightforward.

Furthermore, in a true many-to-many negotiation situation, each of the opponents may be negotiating with a number of competitors to our agent. Since we are not interested in the performance of these competitors, we simulate them by including a break-off probability. This resembles the way outside options are modelled in [6]. However, note that, in contrast to [6], we simulate the actual opponents in our evaluation, and only our opponents have probabilistic outside options which are not actual agents. The result of this break-off probability is that any of the opponents may leave the negotiations before the deadline, simulating their agreement with one of our competitors.

We model the probability of break off using a time-invariant function. In more detail, at any time in the negotiation, the break-off probability during a future time period is given by a function which depends only on the length of that future period. We achieve this by using an exponential function to calculate the probability that an opponent continues to negotiate. Furthermore, we assume that all opponents have the same probability. Specifically, the continuation probability for a given period is given by:

$$\forall i \in A, P_{c,i}(t_a, t_b) = \alpha^{t_b - t_a} \tag{10}$$

where $t_a, t_b > t_a$ are respectively the start and end of the period, and α is a constant which determines the rate of break off. In our experiments, we set $\alpha = 1/n$, where n is the total number of opponents. This ensures that, on average, there will be one agent remaining in the negotiation by the deadline.

4.2 Evaluation Scenarios

We initially evaluated the agents in all scenarios used in the ANAC2011 competition [4]. However, we found that most of these scenarios were not very competitive, and it was often easy to reach agreements with a high utility (for both sides), even using a very simple strategy. This then becomes even easier in a one-to-many negotiation setting against a range of opponents, as it only takes one weak (concessive) opponent to allow any strategy to reach a good agreement. As a result, such scenarios fail to offer sufficient challenge in a concurrent negotiation setting. To address this shortcoming, we selected the largest three scenarios (in terms of the size of their outcome spaces) from the previous two competitions, specifically, the *Travel* scenario from ANAC2010 and the *Energy* and *ADG* scenarios from ANAC2011. Moreover, to make them more competitive, we ensure that the preferences of both parties are strictly opposing. That is,

$$\forall v_{i,x}, v_{i,y} \in V, U_{a,i}(v_{i,x}) \leq U_{a,i}(v_{i,y}) \Leftrightarrow U_{b,i}(v_{i,x}) \geq U_{b,i}(v_{i,y}) \tag{11}$$

To generate a variety of scenarios, we choose the values for each issue by sampling from a uniform distribution, and sorting them such that the strict opposition constraint in Equation 11 is satisfied. Furthermore, the weights for each issue are also sampled from a uniform distribution, normalised such that they sum to one. Since we can generate any number of scenarios using this approach, we refer to the underlying characteristics as the *scenario type*. The details of the scenario types are given in Table 1. We note that, in each particular negotiation, all opponents had the same preferences, but were using a different strategy (as described below).

Table 1. Characteristics of different scenario types.

Name	Number of issues	Number of values for each issue	Number of potential outcomes
Energy	8	5	390,625
Travel	7	4-8	188,160
ADG	6	5	15,625

Furthermore, in order to ensure that decommitment is a viable option for the participants, but is not completely free, we set $D = 0.1$. Moreover, in each negotiation, there is a deadline of 3 minutes, which is common to all participants.

4.3 Evaluation Benchmarks

We tested our agent using a state-of-the-art agent and a very simple agent as benchmarks. In more detail, as the state-of-the-art agent, we use the strategy developed by Nguyen [7]. A limitation of this strategy is that it requires prior knowledge about the payoffs of various strategies against different opponent classes (i.e., tough, linear, and conceder). To determine these values in a principled manner, we used the results from a set of negotiations between simple time-dependent strategies, in a bi-lateral negotiation setting. In more detail, we ran many negotiations between tough, linear and conceder strategies, in all ANAC2011 domains, averaging the results across those domains in order to produce the payoff matrix required by Nguyen's strategy.

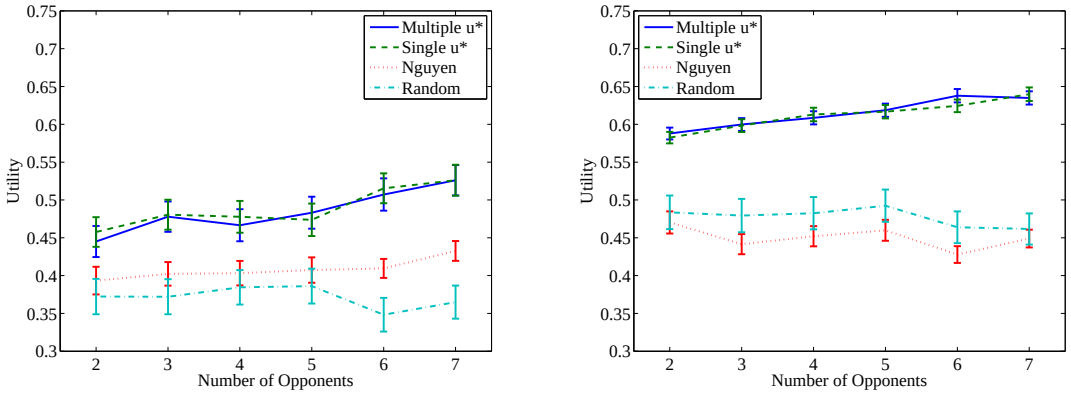
As an additional benchmark, we developed an agent which makes random offers above a fixed threshold (which is chosen randomly in each negotiation session).

4.4 Evaluation Results

For each of the four agents (the two variants of our strategy, Nguyen's strategy, and the random strategy), we run experiments with different numbers of opponents. Each experiment consists of 105 different negotiations per scenario type (totalling 315 negotiations per value of n and per agent). In each negotiation, any opponent strategy from the ANAC competition appears at most once. At the same time, we select the set of opponent strategies in a particular negotiation such that they are equally represented within the experiment. For example, since we have 7 different opponents, if $n = 3$, each opponent appears in exactly 45 out of 105 negotiations. If $n = 7$, then each opponent appears in all negotiations, etc. The value of 105 was chosen since it allows for equal representation for any $n \in [2, 7]$.

The results of these tests are shown in Figure 3, averaged over all scenarios. The error bars show the 95% confidence intervals. More specifically, Figure 3(a) shows the average utilities achieved across all negotiations, including those in which no agreement was reached, whereas in Figure 3(b), we exclude negotiations which did not lead to an agreement. The error bars in Figure 3(a) are large, since these results include a considerable number of outcomes in which no agreement is reached (and therefore have a utility of zero). In contrast, the error bars in Figure 3(b) are smaller since they exclude the disagreement outcomes. Both figures show that, for all numbers of opponents, our strategy achieves a higher average utility than both Nguyen's strategy and the random strategy. In Figure 3(a) our agent achieves a utility between 13% and 26% higher than Nguyen's strategy, increasing with the number of opponents. When considering the utility only in negotiations where agreements were reached (Figure 3(b)), this improvement ranges from 25% to 49%. If we consider the increase

[7] We exclude IAMhaggler2011 as an opponent, since our strategy is based on that strategy.



(a) All negotiations.

(b) All negotiations which resulted in an agreement being formed.

Figure 3. Average results, according to the number of opponents, for a range of different strategies. Error bars indicate the 95% confidence intervals.

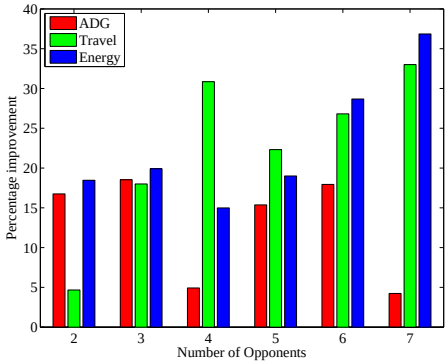


Figure 4. Average utility improvements of our *multiple* u^* strategy over Nguyen's strategy, according to the number of opponents and domain.

in utility separately in each of the three domains (see Figure 4), we observe these results hold in all domains considered. However, the increase is more significant in the domains with larger outcome spaces (the *Travel* and *Energy* domains). Specifically, in the largest domain, with 7 opponents, our strategy achieved a utility 37% greater than that of Nguyen's strategy.

We also considered the situation in which decommitment was strongly discouraged, by making the decommitment penalty infinitely large. The effect of this is a minor reduction in utility achieved (of no more than 0.04), across the different strategies and numbers of opponents. This shows that decommitment has very little impact on the outcome. The random strategy was the one which was most affected by this change, since, due to its potential to make initial agreements with very low utility, the opportunity to make further, better agreements could affect the utility by a greater amount than is possible for a less concessive approach.

Interestingly, we did not find a statistically significant difference between the performance of our two strategies. This suggests that, even though a more complex strategy which allows for a different utility level in each thread could, in theory, achieve a higher utility, in practice a simpler strategy performs equally well. This can be partly explained since the more complex strategy is more computationally expensive, and therefore can generate fewer offers within the same time span compared to the simpler strategy. This is particularly relevant in multi-issue negotiations, where exploring the negotiation space is important. Nevertheless, even though the complex strategy shows no performance gains on average, the additional flexibility could provide benefits in specific domains.

5 CONCLUSIONS AND FUTURE WORK

This paper proposes a novel agent-based strategy for concurrent negotiation against unknown opponents in complex, real-time domains. We formulate our strategy as an optimisation problem under uncertainty, where the decisions are based on probabilistic information about the opponents acquired during the negotiation. Our method coordinates decisions and computes optimal target utility levels across multiple bilateral negotiation threads. We validate our approach against a set of benchmarks, and we show that it outperforms the state-of-the-art in this field [7] by a significant amount, which increases as the number of concurrent negotiation opportunities and the size of the outcome space increases.

In future work, we plan to study in more detail the issue of decommitment, such as adapting our optimisation strategy to also take into account the utility of future decommitment decisions. Moreover, the computational experiments focus on the one-to-many aspect of the negotiation, modelling the outside options of the opponents simply as probability distributions. In future work, we plan to simulate the many-to-many aspect of complex negotiations more directly. Finally, we see this work as a potential contribution towards the development of a specialised one-to-many negotiation track at the international negotiating agents competition (ANAC).

REFERENCES

[1] S. Aknine, S. Pinson, and M. F. Shakun, 'An extended multi-agent negotiation protocol', *Aut. Agents & Multi-Agent Syst.*, **8**(1), 5–45, (2004).
[2] B. An, N. Gatti, and V. R. Lesser, 'Extending alternating-offers bargaining in one-to-many and many-to-many settings', *Proc. of IEEE/ WIC/ACM Conf. on Intelligent Agent Technology*, 423–426, (2009).
[3] B. An, K.M. Sim, L.G. Tang, S.Q. Li, and D.J. Chen, 'Continuous-time negotiation mechanism for software agents', *IEEE Transactions on Systems, Man, and Cybernetics, (Part B)*, **36**(6), 1261–1272, (2006).
[4] T. Baarslag, K. Hindriks, C. M. Jonker, S. Kraus, and R. Lin, 'The second automated negotiating agents competition (ANAC 2011)', *Studies in Computational Intelligence (to appear), Springer*, (2012).
[5] K. Hindriks, C. M. Jonker, S. Kraus, R. Lin, and D. Tykhonov, 'GENIUS: negotiation environment for heterogeneous agents', *Proc. 8th Int. Conf. on Aut. Agents and Multiagent Syst.*, **2**, 1397–1398, (2009).
[6] C. Li, J. A. Giampapa, and K. P. Sycara, 'Bilateral negotiation decisions with uncertain dynamic outside options', *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, **36**(1), 31–44, (2006).
[7] T. D. Nguyen and N. R. Jennings, 'Managing commitments in multiple concurrent negotiations', *Electronic Commerce Res. and Appl.*, **4**, 362–376, (2005).
[8] A. Wächter and L. T. Biegler, 'On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming', *Mathematical Programming*, **106**, 25–57, (2006).
[9] C. R. Williams, V. Robu, E. H. Gerding, and N. R. Jennings, 'Using gaussian processes to optimise concession in complex negotiations against unknown opponents', *Proc. of the 22nd Int. Joint Conf. on Artif. Intell.*, **1**, 432–438, (2011).