Attila: a Negotiating Diplomacy Player Based on Purely Symbolic A.I.

Dave de Jonge[1] [0000−0003−2364−9497] (corresponding author)
and Laura Rodriguez Cima[1] [0009−0001−7630−4102]

IIIA-CSIC, Bellaterra (Barcelona), 08193, Spain
{davedejonge,laura.rodriguez}@iiia.csic.es

Abstract. The board game Diplomacy is considered one of the most challenging test cases for automated negotiation. While many bots have been developed for this game, very few of them are able to negotiate successfully, and the ones that do, have been trained on large data sets of human example games. This makes it hard to apply the same techniques to other games or negotiation scenarios for which no human knowledge is (yet) available. Furthermore, since those bots were trained using deep learning, they are essentially black-boxes for which it is hard to understand how they work. So, these bots do not help us much in gaining a better understanding of strong negotiation techniques. Therefore, in this paper we present a new Diplomacy bot, called Attila, that is purely based on symbolic A.I. Its negotiation algorithm makes use of an existing oracle for the tactical part of the game, called the 'D-Brane Tactical Module' (DBTM). We explain how the DBTM can be converted into a search algorithm for automated negotiation, and we present experiments that show that Attila strongly outperforms several state-of-the-art Diplomacy bots.

1 Introduction

As the use of intelligent agents is becoming increasingly widespread, it is to be expected that in the near future it will be common for multiple such agents to be deployed to work together, with each of them representing a different stakeholder with different objectives. This would require negotiation between such agents, and we therefore expect that automated negotiation will become a more and more prevalent technology in our daily lives. In order to develop such technology we need challenging test cases. One such test case that has been widely studied is the game of Diplomacy.

Diplomacy is a board game for seven players that is also widely played over the Internet. Similar to other board games such as chess and go, it has completely observable game states,[1] it does not involve any random events, and it has an extremely large search space with a branching factor of about 34^{16} possible joint moves per turn [20]. However, the main difference between Diplomacy and other classical board games, is that the players may form coalitions and may negotiate with each other to make agreements

[1] One could argue that Diplomacy does have hidden information, because players make secret agreements. However, these agreements are part of the players' strategies rather than of the game state.

about which moves they will make. One can therefore say that Diplomacy consists of two 'layers'; a *tactical layer*, and a *negotiation layer*. The tactical layer involves the players moving their pieces over the board, while the negotiation layer involves the players negotiating with each other about their moves. Good players need to master both aspects of the game.

Although Diplomacy has been studied for a long time, most bots that have been developed for this game are not able to negotiate [23,8,26,27]. A major breakthrough, however, was made recently by a team of researchers from Meta, who implemented a negotiating Diplomacy bot that was able to beat human players [5]. Their approach used a deep-learning model that was trained on a database of games played by humans. Similarly, a team from DeepMind also implemented a successfully negotiating bot, trained on human data [18]. While the key challenge of Diplomacy has now been tackled using deep learning, this approach has two downsides. Firstly, as is often the case with machine learning, the aforementioned bots are essentially black-boxes, which makes it very hard to tell *why* they are successful. They therefore do not help us much in gaining more insight into the topic of automated negotiation.[2] Secondly, both of these bots required a large corpus of human data in order to be trained. This means that one cannot follow the same approach to apply negotiation to other games or use cases for which no such data is (yet) available.

We therefore present a new negotiating Diplomacy player, called Attila (Advanced negoTiaTing dIpLomAcy bot), that is based on purely symbolic A.I. techniques. For its implementation we followed an approach that was suggested in [12]. That is, we used an existing oracle for the tactical layer of the game, called the D-Brane Tactical Module (DBTM) and applied a trick that allowed us to use it for the negotiation layer as well. This trick consists in pretending that we are controlling the armies of two different players (the one played by our agent, plus its coalition partner), and asking the DBTM to find the best moves for this combined player. These moves can then be proposed to the coalition partner as a potential deal. This idea may seem simple, but, to the best of our knowledge, no one else has tried this approach in the game of Diplomacy before.

While the implementation of our bot is entirely specific to Diplomacy, we think the underlying ideas can be generalized to other games or negotiation scenarios, as long as one has access to some algorithm to determine good actions for an individual non-negotiating agent. For example, in [13] a scenario was described in which logistics companies negotiated to exchange their truck loads. In this scenario one can use the same approach by pretending all truck loads belong to the same company, and then using a standard optimization algorithm to find the optimal solution for that one company.

Of course, our approach does require knowledge of the opponents' utility functions, but we argue that in many realistic scenarios one does indeed have at least approximate knowledge about the opponent's preferences, and otherwise such knowledge can be obtained at run-time using opponent modeling algorithms. While this knowledge may not be perfectly accurate, it can still be used to apply our approach and any other negotiation algorithm would also have to work with such approximate knowledge anyway.

[2] One could try to gain such insight using techniques from Explainable A.I., but to the best of our knowledge this has not been done for those bots.

Another advantage of our approach is that it is entirely transparent. We do not only know that it works, but we also understand *why* it works, which may be useful for the design of new negotiation algorithms for other domains. Finally, another elegant feature of Attila is that it does not depend on any parameters that need to be fine-tuned.

We should stress that *the aim of our research is purely to investigate negotiation algorithms for highly complex scenarios*, so *we do not aim to build a full-fledged Diplomacy player that can also handle all other aspects of Diplomacy*, such as trust, coalition formation, or human psychology, and Attila is not intended for play against humans. For this reason our research focuses only on a variant of Diplomacy in which all agreements are binding. That is, once players make an agreement, they have to obey it.

The source code of Attila has been made publicly available at:
`https://www.iiia.csic.es/~davedejonge/downloads`

2 Related Work

The topic of automated negotiation has been studied as early as the 1950's [22], but mainly from a theoretical point of view. The study of negotiation *algorithms* took off in the 1990's [6]. However, most negotiation algorithms that have been published since, require the agent to calculate the utility value of every possible agreement. Such algorithms are not feasible for many real-world scenarios in which the number of potential deals is astronomically large, or in which the calculation of the utility values is computationally hard. For such scenarios automated negotiation needs to be combined with intelligent search algorithms to search for the best deals to propose. While several papers have been published on this topic [9,14,15,13,17], it still remains an under-investigated problem. Especially, the game of Diplomacy has been identified as an ideal test bed to experiment with such algorithms [4].

The application of A.I. to the game of Diplomacy dates back to 1989 [19,20]. However, most Diplomacy bots that were developed in the early days were not capable of negotiation [23,8,26]. One of the most notable of these bots is DumbBot, which played surprisingly well, given that it was based only on a few simple heuristics. For a long time, DumbBot was the default benchmark used to compare new Diplomacy bots. Several negotiating bots have been shown to outperform the DumbBot, while being equal or similar to the DumbBot if they did not negotiate. Examples are Shaheed's Diplomat [25], the Diplominator [28], Fabregues' bot [3], and DipBlue [7].

To make the development of Diplomacy bots easier for scientific research, the DipGame framework was developed [4], which was later extended into the BANDANA framework [16]. BANDANA included a new bot called D-Brane, which was much stronger than DumbBot. It was shown that, even if D-Brane does not negotiate, it would still outperform some of the bots mentioned above that did negotiate [16].

One important aspect of D-Brane, is that it consists of two separate modules, namely a *tactical* module and a *negotiation* module, which respectively focus on the two different layers of the game. Its tactical module (the DBTM) was made publicly available to allow researchers to implement new negotiation algorithms on top of it. This allowed researchers to focus purely on the implementation of negotiation algorithms, and compare several such algorithms, without having to care about the tactical aspects of the

game. Since then, many new bots have been developed by implementing a negotiation algorithm on top of the DBTM, but none of them resulted in a very clear increase of performance with respect to the non-negotiating D-Brane. One such agent, called AlphaDip [21], did show some improvement, but the authors still concluded that their negotiation algorithm only had a very small influence on its performance.

The DBTM was also used in the Diplomacy Challenge that was part of the Automated Negotiating Agents Competition (ANAC) from 2017 till 2019. However, again, none of the algorithms that were submitted to this competition managed to clearly improve over the non-negotiating D-Brane [12,1]. Therefore, the question how to implement a negotiation algorithm for Diplomacy (and for complex negotiation scenarios in general) remains an important open problem.

3 Automated Negotiation

The research field of automated negotiation deals with multi-agent systems in which the agents are purely self-interested and have opposing interests, but nevertheless still need to cooperate in order to find mutually beneficial solutions. That is, the agents need to jointly come to some agreement by exchanging proposals to one another, according to some negotiation protocol that defines when the agents are allowed to propose or accept which offers, and when such an offer becomes a formally binding agreement [24].

In general, in automated negotiation one assumes there is a finite set of agents $\alpha_1, \alpha_2, \ldots \alpha_n$, and a set of potential **offers** Ω that the agents may propose to each other. Each agent α_i has its own **utility function** $u_i : \Omega \to \mathbb{R}$ that maps each offer $\omega \in \Omega$ to a utility value $u_i(\omega) \in \mathbb{R}$. Furthermore, one usually assumes there is a given deadline T. If the agents do not come to an agreement before this deadline, the negotiations fail. The utility value rv_i that each agent α_i receives in that case, is called the **reservation value**. This means that a rational agent would never accept any proposal that yields less utility than its reservation value. We therefore say that an offer ω is **rational for agent** α_i if $u_i(\omega) \geq rv_i$ and we say that ω is **individually rational**[3] iff it is rational for every agent involved in that offer and for at least one of those agents α_i we have $u_i(\omega) > rv_i$.

Definition 1. *A **negotiation scenario** $\mathcal{N}$ consists of a a finite set of agents $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, a set of offers Ω, for each agent α_i a utility function $u_i : \Omega \to \mathbb{R}$, for each agent α_i a reservation value $rv_i \in \mathbb{R}$, a negotiation protocol, and a deadline T.*

4 Diplomacy

We here give a short description of the game of Diplomacy. It can be formalized as an extensive-form game with seven players, who make their moves simultaneously. The full set of rules is rather complex, so we only give a simplified description.[4]

[3] This term may be somewhat confusing since it refers to rationality for *all* agents, but this is standard terminology in the literature.

[4] We refer to `https://www.wizards.com/avalonhill/rules/diplomacy.pdf` for a complete description of the rules.

4.1 The Tactical Layer

The seven players of Diplomacy are called *England*, *Russia*, *Germany*, *France*, *Turkey*, *Austria* and *Italy*, which are usually abbreviated to ENG, RUS, GER, FRA, TUR, AUS, and ITA. The game takes place on a map of europe which is divided into 75 **provinces**. We denote the set of players as $\mathcal{A}$ and the set of all provinces as $\mathcal{P}$.

Each player begins with 3 or 4 armies that are each placed in a different province. At any time during the game any province can hold at most one army. Therefore, a **game state** s can be formalized as a map $s : \mathcal{P} \rightarrow \mathcal{A} \cup \{\epsilon\}$ that maps each province to a player that is currently occupying the province, or to the symbol ϵ representing that the province is empty. We denote the set of all possible game states as $\mathcal{S}$.

Some of the provinces (34 of the 75) are so-called **supply centers** (SC). We say a player becomes the **owner** of a supply center if she moves one of her armies into that SC (she *conquers* the SC). If that player later moves her army out of that SC, she remains the owner, until another player moves into it. If the owner of an SC changes, the new owner will receive an extra army in the next round, and the previous owner will lose one army. A player is eliminated when she loses all her SCs, and thus all her armies. The game ends either when one player becomes the winner by owning 18 or more SCs (a *Solo Victory*), or when all players that have not yet been eliminated agree to a draw.

In each round each player needs to decide what to do with each of her armies. In Diplomacy-terminology we say that each player must *submit an order* for each of her armies. The player can either submit a *move* order, meaning that she tries to move the army from its current province to an adjacent province, or a *support* order, which means the army will not move, but instead will increase the strength of another army. To formalize this, we define an **order** o to be a tuple of the form $(P, move, P')$ (a **move order**), or of the form (P, sup, P') (a **support order**). The order $(P, move, P')$ represents that a player aims to move an army from province P to province P', while (P, sup, P') represents that she wants the army in province P to give support to the army in province P'. We denote the set of all possible orders as $\mathcal{O}$. Given a game state s and a player α, we say an order $(P, move, P')$ or (P, sup, P') is an order **for an army of** α, if $s(P) = \alpha$. Each round the players submit their orders simultaneously, so each player has to decide her orders without knowing the orders submitted by the other players. If an army a in province P is ordered to move to a province P', then it will only succeed if it has more supports than any other army that is also ordered to move to province P'. If the army a does not succeed, then it will either stay in its current province P, or it will be expelled from it, in case there is some other army that successfully invades P.

4.2 The Negotiation Layer

One very important aspect of Diplomacy is the fact that an army of one player α_i may give support to an army that belongs to another player α_j. Therefore, players can help each other to defeat other players. To facilitate such cooperation between players, each round is divided into two stages: a negotiation stage followed by an action stage. During the action stage the players submit their orders, while during the preceding negotiation stage the players can negotiate with each other, in private, about which orders they

will (or will not) submit during the action stage. Therefore, each game state s can be identified with a negotiation scenario $\mathcal{N}_s$. The agents in this scenario are just the seven players of the game. An offer ω is a set of orders (this set may contain orders for armies of any of the players), so we have[5] $\omega \subset \mathcal{O}$ and $\Omega = 2^{\mathcal{O}}$. Whenever some players come to an agreement ω, it means that each of those players agrees to submit the orders in ω for their respective armies.

Formalizing the utility functions and reservation values of $\mathcal{N}_s$ is harder, because in Diplomacy there is no closed-form expression for a function that maps agreements to utility values. This is because the true utility of a player is not determined by a single agreement, but rather by the final outcome of the game, which depends on *all* the agreements made throughout the entire game, as well as on the moves that all the players make. Therefore, we can only *estimate* the utility values and reservation values of $\mathcal{N}_s$ heuristically. In the rest of the paper we will use the notation $\overline{u}_i(\omega)$ to denote the *estimated* utility (according to some heuristic), for player α_i and offer ω, and $\overline{rv}_i$ to denote the estimated reservation value of α_i.

For the negotiation protocol of $\mathcal{N}_s$, we use the *unstructured negotiation protocol* [14], because it is the standard protocol used by the BANDANA framework, and is most similar to the way humans negotiate in a real Diplomacy game. In this protocol, players do not have to take turns. Instead, each player can propose or accept any offer at any time. A proposal may involve any number of players and it is sent to all the players involved in it, while it remains secret to all other players. A proposal becomes officially binding when accepted by all players involved in it.

It is important to note that, in a real Diplomacy game, agreements between players are purely informal. That is, players are not obliged to obey the agreements they make, and each player must make a personal judgment about whether she thinks the other players will obey their parts of the agreement. While this is a very important aspect of Diplomacy, the goal of our work is not so much to develop a full-fledged Diplomacy player, but rather to investigate automated negotiation techniques. We therefore assume a variant of the game in which all players are obliged to obey their agreements.

5 The D-Brane Tactical Module

We here give a short description of the DBTM and how it can be used to estimate reservation values and utility values.

5.1 Formalization of the DBTM

The DBTM is an algorithm that calculates two functions τ_{sc} and τ_o that we define below. It should be noted, however, that due to the complexity of the game it is not always able to calculate them exactly, but in most cases it does make a very accurate

[5] For the sake of simplicity we are ignoring two important facts here. Firstly, BANDANA also allows players to propose *demilitarized zones*, rather than orders, but this is not relevant to our paper. Secondly, not every set of orders obeys the rules of the game. However, we could still allow players to *propose* such illegal sets of orders. They will just not have any effect when they are submitted.

approximation. The functions τ_{sc} and τ_o each take as their input a triple (s, α, ω_{in}), where s is a game state, α is a player, and $\omega_{in} \subset \mathcal{O}$ is a (possibly empty) set of orders. The function τ_{sc} outputs a non-negative integer $sc \in \mathbb{N}$ and the function τ_o outputs a set of orders $\omega_{out} \subset \mathcal{O}$. So, we have:

$$\tau_{sc} \ : \ \mathcal{S} \times \mathcal{A} \times 2^{\mathcal{O}} \ \rightarrow \ \mathbb{N} \qquad \text{and} \qquad \tau_o \ : \ \mathcal{S} \times \mathcal{A} \times 2^{\mathcal{O}} \ \rightarrow \ 2^{\mathcal{O}}$$

To explain their meaning, let us first focus only on the case that $\omega_{in} = \emptyset$. In this case, the number $\tau_{sc}(s, \alpha, \emptyset)$, represents the number of SCs that player α would own in the next state of the game, when the current state is s, while α submits the set of orders that maximizes the number of SCs for α, and while all other players submit those orders that minimize this number. In other words, it is the 'paranoid' maximin value for α, w.r.t. the number of SCs owned at the end of the turn.

The set $\omega_{out} = \tau_o(s, \alpha, \emptyset)$ contains exactly one order for each army of α, and no other orders. It represents the set of orders corresponding to the output of τ_{sc}. That is, it is the set of orders that α should submit in order to maximize the number of SCs it will own, under the paranoid assumption that all other players try to minimize it. The non-negotiating D-Brane agent always submits exactly these orders.[6] It should be noted that the return value of τ_o is not always uniquely defined, because there could be multiple sets of orders that satisfy this definition. In that case, the set ω_{out} returned by the DBTM depends on the implementation details of the DBTM.

In case that $\omega_{in} \neq \emptyset$, the value $\tau_{sc}(s, \alpha, \omega_{in})$ is a *constrained* maximin value. That is, it is the maximin value under the constraint that at least the orders in ω_{in} are submitted. The set ω_{in} may contain orders for any player, so it may constrain the actions of α itself, as well as the actions of any other player. However, ω_{in} cannot include more than one order for the same army. Similarly, the set of orders $\omega_{out} = \tau_o(s, \alpha, \omega_{in})$ represents those orders that α should submit in order to be guaranteed at least its maximin value, under the constraint that at least the orders in ω_{in} are submitted. This implies that whenever there is an order in ω_{in} for an army of α itself, then that same order must also be present in ω_{out}. The main purpose of calculating τ_o with non-empty ω_{in}, is that it allows a player to find the best set of orders when some of the orders are already fixed by a negotiated agreement ω_{in}. The calculation of τ_{sc} with non-empty ω_{in} can be used to predict how many SCs a player would gain from a potential agreement ω_{in}.

5.2 Estimating Utility and Reservation Values

Given any game state s, we can use the DBTM to estimate the reservation values of the corresponding negotiation scenario $\mathcal{N}_s$ as follows:

$$\overline{rv}_i := \tau_{sc}(s, \alpha_i, \emptyset) \tag{1}$$

Furthermore, when players α_i and α_j agree to submit the set of orders ω, then α_i's estimated utility value $\overline{u}_i(\omega)$ for that deal can be calculated as:

$$\overline{u}_i(\omega) := \tau_{sc}(s, \alpha_i, \omega) \tag{2}$$

[6] More precisely: it submits the orders calculated by the DBTM that approximate the theoretically correct set of orders ω_{out}.

6 ATTILA

We are now ready to present Attila. It consists of a negotiation algorithm added on top of the non-negotiating D-Brane. For simplicity we first explain how it negotiates with only one single opponent. Next, we explain how it can negotiate with multiple partners. Attila was implemented in Java, on the BANDANA framework (version v2.3).

Negotiation algorithms are typically implemented according to the BOA model [2]. That is, they are composed of a Bidding strategy, an Opponent modeling strategy, and an Acceptance strategy. Recently, however, it was argued [13] that for complex scenarios such as Diplomacy, this model should be extended to a BOASE model, where the S stands for 'Search', and the E for 'Evaluation'. This means that a negotiating agent should also be composed of an evaluation algorithm that calculates (or estimates) the utility value $u_i(\omega)$ for any given offer ω, as well as a search algorithm that explores the offer space to find good offers to propose. The evaluation component of Attila is the DBTM, as explained in Section 5, and its search component is explained in Sections 6.1 and 6.3. The bidding and acceptance strategies are explained in Section 6.5. Attila does not use any separate algorithm for opponent modeling.

6.1 Finding a Good Proposal

The DBTM was designed to be used only for the tactical layer of the game, so it only returns a set of orders for *one* given player. However, in the case of negotiations, we want to find a set of orders for two or more players. We here show how we can apply a simple trick to allow the DBTM to do that.

Let us assume that Attila plays the role of player α_i, and that it aims to negotiate a deal with player α_j. Furthermore, suppose that s denotes the current game state. In order to find a good offer to propose to α_j, Attila will ask the DBTM for the best possible set of orders for player α_i in a hypothetical alternative game state s'. This alternative game state is identical to s, except that all armies that in reality belong to α_j, now belong to α_i. Formally, for every province P, the state s' satisfies:

$$s'(P) := \begin{cases} \alpha_i & \text{if } s(P) = \alpha_j \\ s(P) & \text{otherwise} \end{cases} \tag{3}$$

So, assuming that Attila has not made any agreements for the current turn yet, it will use the DBTM to calculate $\tau_o(s', \alpha_i, \emptyset)$ and $\tau_{sc}(s', \alpha_i, \emptyset)$. Let us define:

$$\omega^* := \tau_o(s', \alpha_i, \emptyset) \qquad sc^* := \tau_{sc}(s', \alpha_i, \emptyset)$$

Note that ω^* is a set of orders for exactly those armies that belong to α_i in the *hypothetical* state s'. This corresponds to a set of orders for all armies of α_i and α_j combined in the *real* game state s. Specifically, it is the 'best' possible set of orders for those two players, in the sense that there is no set of orders that guarantees more supply centers to the two players combined. Similarly, sc^* represents the number of supply centers that α_i and α_j *combined* are guaranteed to be able to obtain in the real game

state s. Note that in general this number is larger than simply the sum of the number of SCs they could obtain individually:

$$\tau_{sc}(s', \alpha_i, \emptyset) \;\; \geq \;\; \tau_{sc}(s, \alpha_i, \emptyset) + \tau_{sc}(s, \alpha_j, \emptyset)$$

This is because in the calculation of $\tau_{sc}(s', \alpha_i, \emptyset)$ the DBTM takes into account that the armies that in reality belong to α_i and α_j may support each other, while such support orders between different players are not considered when calculating the individual values $\tau_{sc}(s, \alpha_i, \emptyset)$ and $\tau_{sc}(s, \alpha_j, \emptyset)$.

After determining ω^*, Attila uses the DBTM again to calculate $\overline{rv}_i$ and $\overline{u}_i(\omega^*)$ respectively, using Eqs. (1) and (2), with the real game state s. This is to check that ω^* is individually rational. If this is the case, then it can be argued that ω^* is indeed the best possible proposal, since it has been shown that even for purely self-interested agents the optimal strategy is to aim for a deal that maximizes the sum of the utilities of the two agents (among those that are individually rational) [11] .

6.2 Example

Imagine a miniature version of Diplomacy with only three players TUR, RUS, AUS and only three provinces: Trieste (Tri), Budapest (Bud), and Vienna (Vie). Each of these provinces is adjacent to the other two, and each of them is an SC. Now, suppose the game is in a state s where each player is occupying exactly one of those SCs:

$$s(Tri) = TUR, \quad s(Bud) = RUS, \quad s(Vie) = AUS$$

Furthermore, suppose that Attila plays as TUR and aims to negotiate with RUS. Since each of these players only has a single army to attack and each player has one army to defend their SC, neither of the players is strong enough to conquer any of the other SCs by itself. So, if we use the DBTM to calculate their reservation values we get:

$$\overline{rv}_{tur} = \tau_{sc}(s, TUR, \emptyset) = 0$$
$$\overline{rv}_{rus} = \tau_{sc}(s, RUS, \emptyset) = 0$$

This is because for any player α the DBTM will make the paranoid assumption that its two opponents will work together and therefore take α's supply center.

Now, in order to find an offer to propose to RUS, Attila will internally construct the following hypothetical worldstate s':

$$s'(Tri) = TUR, \quad s'(Bud) = TUR, \quad s'(Vie) = AUS$$

In this hypothetical state TUR has two armies adjacent to Vie, so TUR is strong enough to conquer Vie. If we now apply the DBTM to s' and TUR, we get:

$$\tau_{sc}(s', TUR, \emptyset) = 3$$
$$\tau_o(s', TUR, \emptyset) = \{(Tri, move, Vie)\ ,\ (Bud, sup, Tri)\}$$

That is, the DBTM suggests that TUR's army in Tri should move to Vie, with support from TUR's army in Bud. This would result in a total of three SCs for TUR, because

TUR already owns Tri and Bud, and TUR would successfully conquer Vie, regardless of what orders AUS would submit.

Of course, this is only true in the hypothetical state s'. If these orders are submitted in the actual state s, it would mean that only the order $(Tri, move, Vie)$ would be submitted by TUR, while the order (Bud, sup, Tri) would be submitted by RUS, and as a result TUR would end up with two SCs (Tri and Vie) while RUS would end up with only one SC (Bud). So, Attila uses the DBTM again to calculate the utility values for the real game state and finds:

$$\begin{aligned} \overline{u}_{tur}(\omega^*) &= \tau_{sc}(s, TUR, \omega^*) = 2 \\ \overline{u}_{rus}(\omega^*) &= \tau_{sc}(s, RUS, \omega^*) = 1 \end{aligned}$$

We see that for both players the utility of ω^* is higher than their respective reservation values, so we have found an individually rational deal. While TUR obviously benefits from this deal by receiving support from RUS, note that RUS also benefits from it, because RUS now has the guarantee that TUR will not work together with AUS, and thus that RUS will not lose its SC.

As mentioned above, the return value of τ_o is not always uniquely defined. In this example, the DBTM could have just as well returned the orders (Tri, sup, Vie) and (Bud, mov, Tri). As explained, this depends on the implementation details of the DBTM, and Attila simply uses whatever the DBTM returns.

6.3 Generating More Proposals

We argued above that ω^* is the best possible proposal. However, this is only true under a number of assumptions. Firstly, it only holds if ω^* is individually rational. Secondly, it was based on the assumption that the number of SCs is an accurate measure of 'utility', but we know that in reality it is only a heuristic. And finally, it assumes that the opponent is perfectly rational. For this reason, the opponent may in reality still reject the proposal, and therefore it is essential that we also find other potential deals that could be proposed in case ω^* is rejected.

To find alternative proposals we take the following approach. Let o be any arbitrary order from ω^*, and let $\omega_{sup,o}$ be the subset of ω^* consisting of o and all orders in ω^* that give support to o. Then, the new proposal ω' is defined as $\omega' := \omega^* \setminus \omega_{sup,o}$. That is, the original set of orders ω^*, but with order o and all orders that give support to o removed from it. By repeating this procedure for every order $o \in \omega^*$ we can find several alternative offers. For each of them we then estimate its utility using the DBTM and Eq. (2).

For example, supose we have $\omega^* = \{o_1, o_2, o_3, o_4\}$ with:

$$\begin{aligned} o_1 &= (Tri, move, Vie) & o_2 &= (Bud, sup, Tri) \\ o_3 &= (Hol, move, Bel) & o_4 &= (Spa, move, Por) \end{aligned}$$

So, o_1 is supported only by o_2, while o_2, o_3 and o_4 are not supported by any order. Then:

$$\begin{aligned} \omega_{sup,o_1} &= \{o_1, o_2\} & \omega_{sup,o_2} &= \{o_2\} \\ \omega_{sup,o_3} &= \{o_3\} & \omega_{sup,o_4} &= \{o_4\} \end{aligned}$$

This means Attila would generate the following alternative offers:

$$\omega'_1 = \omega^* \setminus \omega_{sup,o_1} = \{o_3, o_4\} \qquad \omega'_2 = \omega^* \setminus \omega_{sup,o_2} = \{o_1, o_3, o_4\}$$
$$\omega'_3 = \omega^* \setminus \omega_{sup,o_3} = \{o_1, o_2, o_4\} \qquad \omega'_4 = \omega^* \setminus \omega_{sup,o_4} = \{o_1, o_2, o_3\}$$

The idea behind this, is that we already know that ω^* is a reasonably good deal, so we want any alternative proposal to be just a small variation of ω^*. By removing the order o we give the player α that was supposed to submit order o the freedom to do something else, which may make the new offer ω' slightly more beneficial for α, although at the expense that it may be slightly less beneficial to the other player.

6.4 Multiple Negotiation Partners

Above, we have explained how Attila negotiates with one given negotiation partner. However, in Diplomacy the players have to decide during the game with whom they will cooperate, and these partners may change over the course of the game. In order to allow Attila to negotiate with multiple partners, we simply let Attila engage in a separate bilateral negotiation with each individual opponent, and let each of these negotiations run simultaneously.

That is, for each other player α_j we find an initial offer ω^*_j in the way described in Section 6.1. This means that for every opponent α_j we create a separate alternative game state s'_j in which the armies of α_j are assigned to α_j, and then find ω^*_j by calling the DBTM to calculate $\tau_o(s'_j, \alpha_i, \emptyset)$. Next, for each opponent, we create a set of alternative offers in the same way as described in Section 6.3. So, we end up with one set of potential proposals for each opponent. Once it makes an agreement with any of those opponents, it stops all negotiations for that round.

Attila does not build long-term relationships, so, in the next round, Attila again starts a new bilateral negotiation with every other player. It does not care with which of those opponents it has already made any deals before.

The current implementation of Attila is not yet capable of true multilateral negotiations in which it proposes deals involving more than two players. However, Attila can be generalized in a straightforward manner to do so. For example, it could generate an optimal deal ω^*_{jk} that involves two other players α_j and α_k, by creating a hypothetical state s'_{jk} in which all armies of α_j and α_k are assigned to Attila.

6.5 Bidding and Accepting

For the bidding strategy Attila uses a combination of the MiCRO strategy [10] and a time-based strategy [6]. At any time t, Attila may determine, for any opponent α_j, whether or not to make a new proposal to that opponent. It does that based on two criteria, which we call the 'MiCRO criterion' and the 'time criterion'. Attila will only make a new proposal to opponent α_j if both criteria are satisfied. The MiCRO criterion states that the number of unique offers that Attila has proposed to α_j should not be greater than the number of unique offers that α_j has proposed to Attila. This forces the opponent to make concessions. The time criterion states that the utility that Attila would receive from its own next proposal should be greater than its '*aspiration level*'

$f(t)$, where f is some function that decreases over time. This criterion ensures that Attila will initially only propose the most profitable offers.

Attila accepts a proposal from α_j iff it is better for Attila than (or equal to) the next offer that Attila would be proposing to that opponent.

6.6 Tactics

Once Attila reaches an agreement ω with some other player, it uses the DBTM to determine the best set of orders to submit, under the restriction that it must obey the agreement. That is, Attila will submit the orders given by $\tau_o(s, \alpha_i, \omega)$ (where s is the current game state and α_i is Attila). In case that no agreement is made, Attila will, just like D-Brane, submit the orders given by $\tau_o(s, \alpha_i, \emptyset)$. This means that in the absence of any agreements (or negotiation at all) Attila is identical to the non-negotiating D-Brane.

7 Experiments

We will now present the results of our experiments with Attila. All experiments were conducted on a laptop with 11th Gen Intel Core i7-1165G7@2.80GHz CPU, 16 GB RAM, and Windows 11. We made sure that each game was automatically stopped after 80 rounds (i.e. after the Winter 1940 phase) and solo-victories were disabled (so a game would continue even if one of the players had 18 or more SCs). Furthermore, any agreement between players was considered officially binding, so players always had to obey their agreements. Deadlines were set to 3 seconds per round.

7.1 The Benefit of Negotiation

The goal of our first experiment was to determine whether the negotiation algorithm of Attila indeed enables the agent to achieve better results than without negotiations. Since, in the absence of negotiation, Attila is identical to D-Brane, any improvement of Attila over D-Brane must be purely due to the negotiation algorithm.

In order to make this comparison, we played a large number of games with two instances of Atilla and five instances of the non-negotiating D-Brane, and we observed how many SCs the Atillas and the D-Branes would score. Since this highly depends on which two players are being played by Attila, we made sure that each coalition of two players was assigned to the two instances of Attila equally often (there are $\frac{7\times6}{2} = 21$ such coalitions, and we played 200 games for each coalition, so we played $21 \times 200 = 4200$ games). The results are displayed in the right-hand column of Table 1. We see that the two Attilas each score an average of 8.40 SCs per game, while the non-negotiating D-Branes each only score 3.44 SCs.

Furthermore, we wanted to know whether negotiations are really necessary to obtain improved results, or if the same results could also be achieved merely by assuming a ‘peace treaty’ between the two Attilas, meaning that they never attack each other. Therefore, we again ran 200 games for each possible coalition of two players, but without negotiation, and we configured the Attilas to make sure they never attacked each other. The results of this experiment are displayed in the center column of Table 1. We

see that in this case the two Attilas each score an average of 5.86 SCs per game, while the non-negotiating D-Branes each only score 4.46 SCs. While this is still an improvement, it is a lot smaller than the improvement achieved by the fully negotiating Attilas.

We can draw two conclusions from this experiment. Firstly, we conclude that Attila is much better than D-Brane. Secondly, while two agents can already improve their scores simply by agreeing not to attack each other, this effect is only small compared to the improvement achieved by the negotiations.

Finally, we tested the statistical significance of our conclusions. We performed a paired t-test to compare the two results in the center column of the table, another paired t-test to compare the two results in the right-hand column, and finally a Welch t-test to compare the two results in the top row.[7] In all cases the p-value was smaller than 10^{-44}.

Agent	**SCs (Peace)**	**SCs (Nego)**
Attila ($\times 2$)	5.86 ± 0.07	8.40 ± 0.08
D-Brane ($\times 5$)	4.46 ± 0.03	3.44 ± 0.03

Table 1: The number of SCs scored by each agent (averaged per game and agent instance) in two settings with each 2 instances of Attila against 5 instances of D-Brane.

Agent	**SCs**
Attila ($\times 1$)	6.80 ± 0.20
D-Brane ($\times 5$)	4.55 ± 0.08
Gunma ($\times 1$)	3.80 ± 0.12

Table 2: The number of SCs scored by each agent (averaged per game and agent instance), in an experiment with 1 Attila, 1 Gunma, and 5 instances of D-Brane.

7.2 Negotiation with a Different Agent

In our previous experiment we have shown that Attila performs well when it negotiates with a copy of itself. In this section we will show that Attila still performs well when negotiating with some other agent.

For this experiment we used one of the agents that participated in one of the ANAC Diplomacy competitions. Like Attila, these agents all consist of a negotiation algorithm built on top of the non-negotiating D-Brane, so without negotiation they are all identical to D-Brane. For our experiment we chose to use an agent called Gunma, because it outperformed the other agents in the competition of 2018. We also considered some of the other agents that performed well in the ANAC competitions, such as CoalitionBot, Frigate, and Oslo_A. However, as observed in [12] and [1], both Frigate and Oslo_A never accept any proposals, due to bugs in their codes, which makes them useless for our experiments. CoalitionBot, on the other hand, *always* accepts any proposal, which makes any experiment with that agent also rather meaningless.

We ran an experiment in which we played 420 games (10 games for each possible coalition) with one instance of Attila, one instance of Gunma, and five instances of the

[7] Note that the two numbers in the center column are obtained from the same set of games, so we had to perform a *paired* test to compare these numbers. The same holds for the right-hand column. On the other hand, the two numbers in the top row are each obtained from an entirely different set of games, so we had to perform an unpaired test to compare them.

non-negotiating D-Brane. We observed the average number of SCs obtained by each of these agents (per game and per agent instance) and displayed the results in Table 2. A paired t-test showed that Attila was better than D-Brane with $p = 10^{-5}$ and better than Gunma with $p = 10^{-8}$. We see that Attila still benefits from the negotiations, but less than in our previous experiment. We also see that Gunma is actually worse off, so apparently it is agreeing to deals that are not very good for itself. This may also explain why Attila scores lower in this experiment, because negotiations are much harder if your negotiation partner is not making any good proposals.

7.3 Negotiations with Multiple Players

Finally, we tested Attila against multiple negotiating opponents. We therefore ran three more experiments, with 2, 3 and 4 instances of Attila respectively. In each game we also included one non-negotiating D-Brane, and the rest of the field was completed with instances of Gunma and Saitama (another negotiating agent that participated in the ANAC Diplomacy competition). It should be stressed that the agents are anonymous, so Attila does not know which other players are also copies of Attila. The results are displayed in Table 3. We see that in all cases Attila clearly outperforms all other agents. For all three experiments we performed a paired t-test to check that Attila was indeed better than Gunma, and the result was affirmative with confidence levels of $p = 0.018$, $p = 0.012$ and $p = 0.065$ respectively. Similarly, we performed such tests to compare Attila with D-Brane and Saitama, and in all cases the p-level was smaller than $3 \cdot 10^{-3}$.

Note that the score of Attila decreases as the number of instances of Attila increases. This is to be expected, since there is a fixed number of supply centers, so the fewer weak agents, the fewer supply centers the Attilas can steal from the weaker agents.

Agent	**SCs**
Attila ($\times 2$)	7.76 ± 0.75
Gunma ($\times 2$)	4.90 ± 0.68
D-Brane ($\times 1$)	4.22 ± 1.11
Saitama ($\times 2$)	2.22 ± 0.31

Agent	**SCs**
Attila ($\times 3$)	6.93 ± 0.57
Gunma ($\times 2$)	4.08 ± 0.72
D-Brane ($\times 1$)	3.00 ± 0.81
Saitama ($\times 1$)	2.05 ± 0.90

Agent	**SCs**
Attila ($\times 4$)	6.23 ± 0.32
Gunma ($\times 1$)	4.28 ± 1.00
D-Brane ($\times 1$)	3.44 ± 0.73
Saitama ($\times 1$)	1.37 ± 0.31

Table 3: The number of SCs scored by each agent (averaged per game and per agent instance), in three experiments with multiple negotiating agents. Between parentheses the number of instances of each agent.

8 Conclusions

We have presented Attila, a new bot for the game of Diplomacy that consists of a negotiation algorithm built on top of the non-negotiating D-Brane. This is the first negotiation algorithm ever that has been shown to clearly improve the performance of the D-Brane, despite the fact that many other researchers have attempted to do this. Attila does not

rely on any machine learning techniques, which makes it more transparent than other approaches, and allows us to get a better understanding of negotiation. Furthermore, it does not require access to any database of example games, and does not have any parameters that need to be learned or fine-tuned.

Acknowledgments

This work was supported by a Juan de la Cierva - Incorporación research grant (IJC2018-036443-I) and a Ramón y Cajal research grant (RYC2022-035229-I) from the Spanish Ministry of Science and Innovation, by a JAE-INTRO-ICU grant (JAEIntroICU-2021-IIIA-09) funded by the Spanish Scientific Research Council (CSIC), and by grant no. TED2021-131295B-C31 funded by MCIN/AEI /10.13039/501100011033 and the European Union NextGenerationEU/PRTR.

References

1. Aydoğan, R., Baarslag, T., Fujita, K., Mell, J., Gratch, J., de Jonge, D., Mohammad, Y., Nakadai, S., Morinaga, S., Osawa, H., Aranha, C., Jonker, C.M.: Challenges and main results of the automated negotiating agents competition (anac) 2019. In: Multi-Agent Systems and Agreement Technologies - 17th European Conference, EUMAS 2020, and 7th International Conference, AT 2020, Thessaloniki, Greece, September 14-15, 2020, Revised Selected Papers. pp. 366–381. Springer International Publishing, Cham (2020)
2. Baarslag, T., Hindriks, K., Hendrikx, M., Dirkzwager, A., Jonker, C.: Decoupling negotiating agents to explore the space of negotiation strategies. In: Marsa-Maestre, I., Lopez-Carmona, M.A., Ito, T., Zhang, M., Bai, Q., Fujita, K. (eds.) Novel Insights in Agent-based Complex Automated Negotiation, pp. 61–83. Springer Japan, Tokyo (2014)
3. Fabregues, A.: Facing the Challenge of Human-aware Negotiation. Ph.D. thesis, Universitat Autònoma de Barcelona (2012)
4. Fabregues, A., Sierra, C.: Dipgame: a challenging negotiation testbed. Engineering Applications of Artificial Intelligence **24**(7), 1137–1146 (2011)
5. (FAIR)†, M.F.A.R.D.T., Bakhtin, A., Brown, N., Dinan, E., Farina, G., Flaherty, C., Fried, D., Goff, A., Gray, J., Hu, H., Jacob, A.P., Komeili, M., Konath, K., Kwon, M., Lerer, A., Lewis, M., Miller, A.H., Mitts, S., Renduchintala, A., Roller, S., Rowe, D., Shi, W., Spisak, J., Wei, A., Wu, D., Zhang, H., Zijlstra, M.: Human-level play in the game of diplomacy by combining language models with strategic reasoning. Science **378**(6624), 1067–1074 (2022)
6. Faratin, P., Sierra, C., Jennings, N.R.: Negotiation decision functions for autonomous agents. Robotics and Autonomous Systems **24**(3-4), 159 – 182 (1998)
7. Ferreira, A., Cardoso, H.L., Reis, L.P.: Dipblue: A diplomacy agent with strategic and trust reasoning. In: ICAART 2015 - Proceedings of the International Conference on Agents and Artificial Intelligence, Volume 1, Lisbon, Portugal, 10-12 January, 2015. pp. 54–65. SciTePress (2015)
8. Hall, M.R., Loeb, D.E.: Thoughts on programming a diplomat. Heuristic Programming in Artificial Intelligence **3**(9), 123–145 (1992)
9. Ito, T., Klein, M., Hattori, H.: A multi-issue negotiation protocol among agents with nonlinear utility functions. Multiagent Grid Syst. **4**, 67–83 (January 2008)
10. de Jonge, D.: An analysis of the linear bilateral ANAC domains using the MiCRO benchmark strategy. In: Raedt, L.D. (ed.) Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022. pp. 223–229. ijcai.org (2022). https://doi.org/10.24963/ijcai.2022/32

11. de Jonge, D.: A new bargaining solution for finite offer spaces. Applied Intelligence **53**(23), 28310–28332 (2023)
12. de Jonge, D., Baarslag, T., Aydoğan, R., Jonker, C., Fujita, K., Ito, T.: The challenge of negotiation in the game of diplomacy. In: Lujak, M. (ed.) Agreement Technologies, 6th International Conference, AT 2018, Bergen, Norway, December 6-7, 2018, Revised Selected Papers. Lecture Notes in Computer Science, vol. 11327, pp. 100–114. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-17294-7_8
13. de Jonge, D., Bistaffa, F., Levy, J.: Multi-objective vehicle routing with automated negotiation. Applied Intelligence **52**(14), 16916–16939 (Nov 2022)
14. de Jonge, D., Sierra, C.: NB3: a multilateral negotiation algorithm for large, non-linear agreement spaces with limited time. Auton. Agents Multi Agent Syst. **29**(5), 896–942 (2015)
15. de Jonge, D., Sierra, C.: GANGSTER: an automated negotiator applying genetic algorithms. In: Fukuta, N., Ito, T., Zhang, M., Fujita, K., Robu, V. (eds.) Recent Advances in Agent-based Complex Automated Negotiation, pp. 225–234. Studies in Computational Intelligence, Springer International Publishing (2016)
16. de Jonge, D., Sierra, C.: D-Brane: a diplomacy playing agent for automated negotiations research. Applied Intelligence **47**(1), 158–177 (2017)
17. de Jonge, D., Zhang, D.: GDL as a unifying domain description language for declarative automated negotiation. Auton. Agents Multi Agent Syst. **35**(1), 13 (2021)
18. Kramár, J., Eccles, T., Gemp, I., Tacchetti, A., McKee, K.R., Malinowski, M., Graepel, T., Bachrach, Y.: Negotiation and honesty in artificial intelligence methods for the board game of diplomacy. Nature Communications **13**(1), 7214 (2022)
19. Kraus, S., Lehman, D., Ephrati, E.: An automated diplomacy player. In: Levy, D., Beal, D. (eds.) Heuristic Programming in Artificial Intelligence: The 1st Computer Olympia, pp. 134–153. Ellis Horwood Limited (1989)
20. Kraus, S., Lehmann, D.: Designing and building a negotiating automated agent. Computational Intelligence **11**, 132–171 (1995)
21. Marinheiro, J., Lopes Cardoso, H.: Towards general cooperative game playing. In: Nguyen, N.T., Kowalczyk, R., van den Herik, J., Rocha, A.P., Filipe, J. (eds.) Transactions on Computational Collective Intelligence XXVIII. pp. 164–192. Springer, Cham (2018)
22. Nash, J.: The bargaining problem. "Econometrica" **"18"**, 155–162 (1950)
23. Ribeiro, J., Mariano, P., Lopes, L.S.: Darkblade: A program that plays diplomacy. In: Progress in Artificial Intelligence, 14th Portuguese Conference on Artificial Intelligence, EPIA 2009, Aveiro, Portugal, October 12-15, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5816, pp. 485–496. Springer (2009)
24. Rosenschein, J.S., Zlotkin, G.: Rules of Encounter. The MIT Press, Cambridge, USA (1994)
25. Shaheed, J.: Creating a diplomat. Master's thesis, Department of Computing, Imperial College Of Science, Technology and Medicine **180** (2004)
26. Shapiro, A., Fuchs, G., Levinson, R.: Learning a game strategy using pattern-weights and self-play. In: Computers and Games: Third International Conference, CG 2002, Edmonton, Canada, July 25-27, 2002. Revised Papers 3. pp. 42–60. Springer (2003)
27. Theodoridis, A., Chalkiadakis, G.: Monte carlo tree search for the game of diplomacy. In: Spyropoulos, C.D., Varlamis, I., Androutsopoulos, I., Malakasiotis, P. (eds.) SETN 2020: 11th Hellenic Conference on Artificial Intelligence, Athens, Greece, September 2-4, 2020. pp. 16–25. ACM (2020). https://doi.org/10.1145/3411408.3411413
28. Webb, A., Chin, J., Wilkins, T., Payce, J., Dedoyard, V.: Automated negotiation in the game of diplomacy (2008)