Appl Intell (2017) 47:158–177
DOI 10.1007/s10489-017-0919-y

INVITED PAPER

D-Brane: a diplomacy playing agent for automated negotiations research

Dave de Jonge[1,2] · Carles Sierra[1]

Published online: 27 February 2017

Abstract Existing work on Automated Negotiations commonly assumes the negotiators' utility functions have explicit closed-form expressions, and can be calculated quickly. In many real-world applications however, the calculation of utility can be a complex, time-consuming problem and utility functions cannot always be expressed in terms of simple formulas. The game of Diplomacy forms an ideal test bed for research on Automated Negotiations in such domains where utility is hard to calculate. Unfortunately, developing a full Diplomacy player is a hard task, which requires more than just the implementation of a negotiation algorithm. The performance of such a player may highly depend on the underlying strategy rather than just its negotiation skills. Therefore, we introduce a new Diplomacy playing agent, called D-Brane, which has won the first international Computer Diplomacy Challenge. It is built up in a modular fashion, disconnecting its negotiation algorithm from its game-playing strategy, to allow future researchers to build their own negotiation algorithms on top of its strategic module. This will allow them to easily compare the performance of different negotiation algorithms. We show that D-Brane strongly outplays a number of previously developed Diplomacy players, even when it does not apply negotiations. Furthermore, we explain the negotiation algorithm applied by D-Brane, and present a number of additional tools, bundled together in the new BANDANA framework, that will make development of Diplomacy-playing agents easier.

Keywords Automated Negotiation · Diplomacy · Constraint Optimization · Game Theory · Search

1 Introduction

In any multiagent system (MAS) the outcome of the actions taken by one agent may also depend on the actions taken by other agents. These agents may have conflicting goals and, since the other agents may be unknown and may not be benevolent, an agent generally cannot assume that other agents are willing to help without receiving any benefits in return. If each agent simply chooses those actions that are individually best, the outcome can be sub-optimal for each of them, as illustrated by the well-known Prisoner's Dilemma [28]. Therefore, agents in a MAS need to negotiate on what actions each will take, even if those agents are entirely selfish and are not interested in reaching any socially optimal solution. Generally, we can say that if a Nash equilibrium [26] is not Pareto optimal then negotiations allow the agents to reach a more efficient solution by making a binding agreement in which each agent promises not to deviate from the efficient solution.

Automated Negotiations have been studied extensively, but most work focuses purely on the strategy to determine which deals to propose *given* the utility values of the possible deals. A point that has received little attention however, is the fact that in many real-world negotiation settings, for any given proposal a negotiator would first need to spend time determining its utility value before he or she could decide whether to propose, accept, or reject it. In most

✉ Dave de Jonge
davedejonge@iiia.csic.es

Carles Sierra
sierra@iiia.csic.es

1 IIIA-CSIC, Campus de la UAB s/n, 08193, Bellaterra, Spain

2 Western Sydney University, Penrith NSW 2751, Australia

existing work this process of evaluating the proposal is simply abstracted away and it is assumed that this does not require any domain knowledge or reasoning. In such studies the negotiators are usually assumed to know the utility value of any deal instantaneously, or after solving a simple linear equation (see for example [3]). The utility functions of the agent's opponents on the other hand, are often assumed to be completely unknown.

We argue, however, that in real negotiations it is important to have knowledge of the domain and one should be able to reason about it. One cannot, for example, expect to make profitable deals in the antique business if one does not have extensive knowledge of antique, no matter how good one is at bargaining. Moreover, a good negotiator should also be able to reason about the desires of its opponents. A good car salesman, for example, would try to find out what type of car best suits his client's needs, in order to increase the chances of coming to a successful deal, and therefore increase the salesman's own expected utility.

Another point that is rarely taken into account, is that an agent's utility may not always solely depend on the agreements it makes, but may also depend on decisions taken outside the negotiation thread, either by the agent itself or by its opponents. Imagine for example buying a small car which is easy to park and consumes little fuel. This may initially be a great deal. However, if one year later you decide to extend your family and have children, that small car suddenly is not so practical anymore. We see that the value of the car deal has changed as a result of decisions made long after the negotiations had finished. As another example, imagine renting a property to open a restaurant in a street with no other restaurants. This might be a good deal until later several other restaurants also open in that same street, presenting you with so much competition that you can no longer afford the rent. Again, what was initially a good deal, later became a bad deal, only this time as a result of decisions taken by competitors.

For these reasons, we think the game of Diplomacy [9] provides a much more realistic, and therefore more interesting, test bed for Automated Negotiations. Diplomacy is important for Automated Negotiations (and for AI in general) because it includes many of the difficulties one would also have to deal with in real-life negotiations. It involves constraint satisfaction, coalition formation, game theory, trust, and even psychology. Being a good Diplomacy player does not only require strategic insight, but also requires social skills, making it a particularly hard game for computers. It is not surprising therefore, that computer Diplomacy is only in its infancy and automatic players are not nearly as well developed as for example Chess or Go programs. Now that modern Chess computers are already far superior to any human player, we expect that Diplomacy will draw more and more attention in the future, as a more interesting challenge for computer scientists. In line with this expectation in July 2015 the first edition of the Computer Diplomacy Challenge[1] was held as part of the ICGA Computer Olympiad.

In this paper we wish to highlight the fact that Diplomacy satisfies the following three important properties:

1. For any potential deal the calculation of its utility value (for any agent) is a hard, time-consuming problem.
2. An agent's utility does not only depend on the agreements it makes, but also on decisions it makes outside the negotiation thread.
3. Moreover, an agent's utility also depends on decisions made by its opponents outside the negotiation thread.

Another important point is that the number of possible deals in Diplomacy is extremely large, so it would be impossible to exhaustively determine the utility values of all possible deals.

The game of Diplomacy has been under attention of the Automated Negotiations community for a long time. Nevertheless, to date very few really successful negotiating Diplomacy players have been developed. The problem with Diplomacy is that before one can test a negotiation algorithm one first needs to have an agent that can play the strategic part of the game and implementing such player is already a daunting task. Therefore, existing work has focused either on building only non-negotiating players, or on building negotiation algorithms on top of existing (often poorly playing) agents. In this paper however, we introduce a new Diplomacy playing agent, called D-Brane, that is a good strategic player, but that is also capable of negotiation. Moreover, we have decoupled its strategic algorithm from its negotiation algorithm so that they can be studied and reused independently. This will allow new negotiation algorithms in the future to be implemented on top of D-Brane's strategic component. We think that this will mean an important step forward in the research of computer Diplomacy and Automated Negotiations, as it will make it much easier for Automated Negotiations researchers to test algorithms for highly complex domains.

We have performed a number of experiments in which we compare our player against several existing Diplomacy playing agents. The interesting outcome of those experiments is that even if our agent does not apply negotiations it still outplays the existing players, which do apply negotiations. From this we draw the important conclusion that *the success of a negotiating agent may sometimes depend more on the accuracy and efficiency in which it calculates utility values than on the bargaining strategy it applies.*

In short, this paper makes the following contributions to the field of Automated Negotiations:

[1]https://icga.leidenuniv.nl/?page_id=987

- We define a new, more realistic, formal model of negotiations which we call a Negotiation Game.
- We present a strategic Diplomacy player that allows researchers to build negotiation algorithms on top of it.
- We present the negotiation algorithm used by our player.
- We show that in complex domains it can be more important to have an efficient and accurate algorithm to determine utility values of potential deals, than to have a good bargaining strategy.
- We present a new framework, called BANDANA, which consists of a number of tools to make development of Diplomacy players easier.

The rest of this paper is organized as follows: in Section 2 we give an overview of existing work on Automated Negotiations and Diplomacy. In Section 3 we give an informal description of the game of Diplomacy. In Section 4 we define the notion of a Negotiation Game, which puts negotiations into a larger context so that the agents' utility values do not only depend on the agreements made, but also on decisions made after the negotiations. In Section 5 we present the Diplomacy playing agent that we have implemented and in Section 6 we present the results of our experiments with this agent and compare them with the results of a number of other existing Diplomacy agents. In Section 7 we draw our conclusions and discuss future work. In Appendix A we present the BANDANA framework which comprises a number of tools to facilitate the development of Diplomacy playing agents. Finally, in Appendix B we give a formal description of Diplomacy.

2 Related work

Much work has been done on Automated Negotiations, which can roughly be divided into two categories: the *Game Theoretical Approach* and the *Heuristic Approach*.

The Game Theoretical Approach focuses on the theoretical properties of negotiation, such as the existence of equilibrium strategies. A seminal paper in this area is a paper by Nash [25] in which he shows that under the assumption of certain axioms the outcome of a bilateral negotiation is the solution that maximizes the product of the players' utilities. This is known as the *Nash Bargaining Solution*. Many papers have been written afterwards that generalize or adapt some of these axioms. Multilateral versions of the bargaining problem have been studied for example in [2, 21], while a non-linear generalization has been made in [12]. These studies give hard guarantees about the success of their approach, but the downside is that they need to make very strong assumptions about their respective domains, which makes them hard to apply in real-world settings. An example of such an assumption is the existence of a discount factor that reduces the utility of any deal by some known factor depending on the time it takes to come to an agreement. Other examples are the assumption that the negotiation has a fixed number of rounds and the negotiators take turns, or even that the negotiators have perfect knowledge about each others' utility functions. A general overview of such game theoretical studies is made in [31].

In this paper, on the other hand, we apply the Heuristic Approach. The Heuristic Approach focuses on the implementation of algorithms that can negotiate under circumstances where no equilibrium results are known, or where the equilibria cannot be determined in a reasonable amount of time. It is usually not possible to give hard guarantees about the success of such algorithms, but they are more suitable to real-world negotiation scenarios.

However, even in this branch of Automated Negotiations one often still makes many simplifying assumptions. One often assumes that negotiations are only bilateral, that there is only a small set of possible agreements to make, and that the utility functions are given as linear additive functions or can be calculated without much computational cost. Also, most of these studies assume an alternating offers protocol, which is fine for automated agents, but not desirable for negotiations with humans, because with humans there is no guarantee that they will indeed follow the protocol. All these assumptions were made for example in the first four editions of the annual Automated Negotiating Agent Competition (ANAC 2010–2013) [3]. Important examples in this area are [10, 11]. They propose a strategy that amounts to determining for each time t which utility value should be demanded from the opponent (the *aspiration level*). However, they do not take into account that one first needs to find a deal that indeed yields that aspired utility level. They simply assume that such a deal always exists, and that the negotiator can find it without any effort.

Recently, more focus has been given to more realistic negotiation settings. Negotiations with non-linear utility functions, for example, have been studied in [22]. The negotiations are, however, still bilateral, the agreement space is continuous and it is assumed the agreements at least have a known closed-form expression. Also in [17] the utility functions are strictly spoken non-linear over the issues, but they are still linearly additive over *pairs* of issues. Moreover, the approach of [17] requires a trusted mediator, or a trusted 'fair die'.

Domains in which the number of possible deals is very large so that one needs to apply search algorithms to find good deals have been studied for example in [16, 23, 24]. Although their utility functions are non-linear over the vector space that represents the space of possible deals, the value of any given deal can still be calculated quickly by solving a linear equation. It is true, as the authors claim, that in theory any non-linear function can be *modeled* in such a way, but the problem is that in real-world settings utility functions are not always *given* in this way (e.g. there is no known closed-form expression for the utility function over the set of all possible configurations of a chess game). In order to apply their method one would first need to transform the given expression of a utility function into the expression required by their model, which may easily turn out to be an unfeasible task. Therefore, we have taken the idea of non-linear utility functions a step further in [4], where for any proposal the evaluation of its utility value required solving a Traveling Salesman Problem. However, we still assumed that utility values were assigned directly to deals. The utility values did not depend on any decisions made outside the negotiation thread.

Most research on Automated Negotiations is restricted to bilateral negotiations. Work on multilateral negotiations often focuses on developing protocols (e.g. [7, 15]) or on non-selfish negotiations [18]. An example of a negotiation algorithm for selfish, multilateral negotiations is given in [27]. In this study however, a strict separation is made between *buyers* and *sellers*, so a buyer can only come to an agreement with a seller. In real-life negotiations one cannot always make such a distinction. A retailer, for example, sells its products to consumers, but buys them from a wholesaler, so acts both as buyer and seller. Moreover, [27] considers that only one buyer is present, therefore excluding competition between buyers. Furthermore, although multiple sellers are present, they still assume that all agreements are strictly bilateral. Also [1] describes multilateral negotiations in which one buyer negotiates with n sellers, in parallel bilateral negotiation threads, but negotiations are only about the price of a single item. In this paper on the other hand, we do assume multilateral negotiations in which there is no distinction between buyers and sellers, and in which a single deal may involve more than two agents.

An alternative way to subdivide the field of Automated Negotiations is to distinguish between Argumentation Based Negotiations (ABN), and non-Argumentation Based Negotiations. In ABN one assumes that agents are capable of exchanging arguments with one another in order to influence the opponents actions. One agent may for example argue why a certain outcome is unacceptable, or why the opponent should change its mind about a certain proposal. For example [33] describes a model of how the beliefs and behavior of negotiators can be changed via persuasive argumentation. In [32] the authors present “a framework for negotiation in which agents exchange proposals backed by arguments which summarize the reasons why the proposals should be accepted”. A general overview of ABN is provided in [6]. Since argumentation plays an essential role in Diplomacy, we think that this game would also be an excellent test case for ABN. However, in this paper we will not apply ABN. Instead, we will leave this as future work.

Pioneering work on negotiations in Diplomacy was presented in [19, 20]. These papers, however, focus mainly on the modular structure of their agent rather than on the algorithms it applies. It remains unclear how their agent searches through the large space of possible deals and determines what to propose. Moreover, they have only been able to play a very small number of games, as they had to play them with humans, which takes a long time. An informal online community called DAIDE exists which is dedicated to the development of Diplomacy playing agents.[2] Many agents have been developed by this community but only very few are capable of negotiation. In [9] a new platform called DipGame was introduced to make the development of Diplomacy agents easier for scientific research. Several negotiating agents have been developed on this platform such as in [8, 13]. Later on in this paper we compare our agent with the agents presented in these studies.

3 Diplomacy, an informal description

Diplomacy is a popular board game invented in the 1950’s which is nowadays also widely played over the Internet.[3] It is a game for seven players, over multiple rounds, with complete information and no chance moves. In order to play well negotiation is an essential skill. Although for each player the ultimate goal of the game is to defeat all other players, players often form coalitions and agree to end the game in a draw between the members of one coalition once all players outside that coalition have been defeated.

The full set of rules of Diplomacy is rather complex, so we only give a simplified description. The differences with the full set of rules are not relevant to this paper anyway.[4] Furthermore, in this Section we will keep the discussion informal. For a formal definition of the rules we refer to Appendix B.

Diplomacy is a game over multiple rounds. The seven players (also referred to as the seven *Great Powers*) are called England, Russia, Germany, France, Turkey, Austria and Italy, which are usually abbreviated to ENG, RUS, GER, FRA, TUR, AUS, and ITA. Each player begins with 3 or 4 units (also called *armies* or *fleets*) that are placed on a map of Europe in the early 20th century. The map is divided into 75 *provinces*, which can each hold 0 or 1 units. Some of the provinces (34 in total) are marked as *Supply Centers*.

[2] http://www.daide.org.uk

[3] http://www.playdiplomacy.com/

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

A player becomes the *owner* of a Supply Center if he moves one of his units into that Supply Center. If that player later moves his unit out of that Supply Center he remains the owner, until another player moves one of his units into it.

If the owner of a Supply Center changes the new owner will receive an extra unit in the next round, and the previous owner will lose one unit. A player is eliminated when he loses all his units. The game ends either when one player becomes the winner by owning 18 or more Supply Centers (a *Solo Victory*), or when all players that have not yet been eliminated agree to end the game in a draw.

In each round each player needs to decide what to do with each of his units. In Diplomacy-terminology we say that each player must *submit an order* for each of his units. He can either submit a *move-to* order, meaning that he tries to move the unit from its current location to an adjacent province, a *hold* order, meaning that the unit intends to stay in its current location, or a *support* order, meaning that the unit will not move, but instead will give extra strength to a moving or holding unit. A unit u can only support a unit u' that moves to (or holds in) a province p if u is located in a province adjacent to p.

The players all submit their orders simultaneously, which means that each player must decide his orders without knowing which orders the other players are submitting in that round.[5]

If two units of two different players are both ordered to move to the same province (or one of them holds in that province), then only the unit that receives the most supports will indeed move, while the other one will stay in its current province (or will be expelled from it if it was trying to hold). In case two units have an equal amount of support then both units will stay in their current province. When a player moves one of his units into a Supply Center he does not own, he will become the new owner of that Supply Center, and therefore we say the player *conquers* that Supply Center. An interesting aspect of this game is the fact that a unit u of one player may give support to a unit u' that belongs to another player. Therefore, players can help each other to conquer Supply Centers.

The main difference between Diplomacy and purely strategic games like Chess and Checkers is that in Diplomacy players are allowed to negotiate with each other and form coalitions. Each round consists of 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 negotiate about which orders they will (or will not) submit during the action stage. Typically, players agree not to attack each other, or they agree that one player will use some of its units to support a unit of the other player.

When a player α_i tries to move into a Supply Center p, or supports a coalition partner to move into p, we say he is participating in a *battle* for p. In order to win the battle (i.e. to successfully move into p and thus become its new owner) α_i must submit a move-to order for some unit u to move into p (or hold order to stay in p) and α_i or any of his coalition partners may submit any number of support orders to support the unit u. We refer to these orders as the *battle plan* of α_i for province p. Typically, more than one player will try to conquer the same province at the same time, so only the player with the strongest battle plan will succeed. Furthermore, the units of a player are often spread around the map of Europe, so during any round a player may be involved in several battles at the same time.

Example Let us focus on the three players ENG, FRA and GER and suppose that ENG and FRA together form a coalition. These players submit the following orders:

1. ENG moves his unit in the North Sea to Holland.
2. FRA's unit in Belgium supports ENG's unit in the North Sea.
3. GER moves his unit in Kiel to Holland.
4. FRA moves his unit in Burgundy to Munich.
5. GER holds with his unit in Munich.
6. GER's unit in Silesia supports GER's own unit in Munich.

We see here there are two battles going on: a battle for Holland and a Battle for Munich. The first two orders together form a battle plan of the coalition {ENG, FRA} to conquer Holland and the third order is the battle plan of GER to conquer Holland. The fourth order is a battle plan of FRA to conquer Munich, and the fifth and sixth orders form GER's battle plan to defend Munich.

Although ENG and GER both try to move to Holland, only ENG will succeed, because ENG's unit has support from FRA. Furthermore, FRA is trying to expel GER 's unit from Munich, but fails, because FRA 's unit does not have any support, while GER's unit in Munich does have support (in fact, even without support GER's unit would not be expelled from Munich, because FRA and GER would have equal strength).

In the rest of this paper we will consider each round of Diplomacy as a separate negotiation scenario that satisfies the three properties we highlighted in the introduction. Indeed, determining the influence of an agreement on the number of conquered Supply Centers is a complex combinatorial problem. Furthermore, the utility of a player α_i is not directly determined by the agreements it makes, but rather by the orders submitted by α_i, as well as the orders submitted by its opponents. In the following sections we will formalize these properties, by defining the notion of

[5] In a real-life game this is achieved by letting each player first secretly write down his orders on a piece of paper and only once everyone has done so, the orders are revealed.

a Constraint Optimization Game, which captures the first property, and the notion of a Negotiation Game, which captures the second and third properties.

4 Negotiation Games

As explained, in this paper we assume that the agents' utility values depend not only on the agreements they make, but also on the decisions they take outside the negotiation thread. In order to model this formally we define the concept of a Negotiation Game.

The idea of a negotiation game is that the players are playing some game G, but before doing so they have the opportunity to negotiate binding agreements about which actions each player will take. The players' utilities are purely determined by their actions in the game G, but since their choice of actions is partially restricted by the agreements they make, the agreements between the players indirectly influence the players' utility values. The negotiation thread followed by the actual playing of the game G are together referred to as the *Negotiation Game over G* and is denoted as $N(G)$.

Definition 1 A **one-shot game** G consists of a set of **players**, $Pl^G = \{\alpha_1, \alpha_2, ...\alpha_n\}$, for each player $\alpha_i \in Pl^G$ a set of **actions** $\mathcal{M}_i^G$ and for each player $\alpha_i \in Pl^G$ a **utility function** f_i^G which is a function from $\mathcal{M}^G = \mathcal{M}_1^G \times \mathcal{M}_2^G \times ...\mathcal{M}_n^G$ to $\mathbb{R}$. An element μ of the set $\mathcal{M}^G$ is called an **action profile** and its **utility vector** $f^G(\mu)$ is the vector $f^G(\mu) = (f_1^G(\mu), f_2^G(\mu), ...f_n^G(\mu)) \in \mathbb{R}^n$.

We now want to allow the players to negotiate over the actions they take. For this reason, we need to define the concept of a *deal*.

Definition 2 A **deal** x over a one-shot game G is a Cartesian product of nonempty subsets $S_i \subseteq \mathcal{M}_i^G$, one for each player: $x = S_1 \times S_2 \times \ldots S_n$. The set of all possible deals over a game G is called the **agreement space** Agr_G of G.

A deal should be interpreted as an agreement between the players that each of them will only choose its action from the subset S_i.

Example Let G be the Prisoner's Dilemma. There are two players: $Pl^G = \{\alpha_1, \alpha_2\}$ and their actions are: $\mathcal{M}_1^G = \mathcal{M}_2^G = \{\text{confess, deny}\}$. The agreement space Agr_G consists of all products $S_1 \times S_2$ where S_1 and S_2 can be either {confess}, {deny} or {confess, deny}. So Agr_G contains 9 possible deals. The deal {confess} × {confess, deny} for example would represent the agreement that α_1 will play 'confess', while α_2 can play either 'confess' or 'deny'.

Regarding to this example, we should note that in the Prisoner's Dilemma the players are of course not allowed to negotiate. However, this does not mean that we cannot define its Agreement Space. After all, according to Definition 2 a deal is a well-defined concept for any one-shot game G even if players are not allowed to negotiate. This is important because we first need to define the Agreement Space of a non-negotiation game G before we can define the negotiation version $N(G)$ of that game. Therefore, Agr_G should be interpreted as the space of deals that the players *could* make *if they were allowed* to negotiate.

In the following we will use the notation $\mathcal{M}_i^G[x]$ instead of S_i to explicitly indicate that it is part of a deal x. Note that if $\mathcal{M}_i^G[x] = \mathcal{M}_i^G$ player α_i is not affected by the deal; the set of actions he can choose from is not restricted by the deal. We therefore say that an agent is participating in a deal x if $\mathcal{M}_i^G[x]$ is a *strict* subset of $\mathcal{M}_i^G$.

Definition 3 The set of **participating agents** $pa(x)$ of a deal x is defined as:

$$pa(x) = \{\alpha_i \in Pl^G \mid \mathcal{M}_i^G[x] \neq \mathcal{M}_i^G\}$$

Definition 4 Given a one-shot game G and a deal x of Agr_G, the **restricted game** $G[x]$ is the same game as G except that each player α_i is only allowed to choose its action from the subset $\mathcal{M}_i^G[x]$ rather than its full set of actions $\mathcal{M}_i^G$. Similarly, for a set of deals X the game $G[X]$ is the game G with the restriction that each player α_i can only choose its action from the intersection $\bigcap_{x \in X} \mathcal{M}_i^G[x]$.

Definition 5 A set of deals X is called **consistent** iff its intersection is nonempty: $\bigcap_{x \in X} x \neq \emptyset$.

Note that if X is not consistent, it means that there is no action profile that satisfies all agreements in X, so it is impossible to obey them all.

Given a one-shot game G and a positive integer d the *negotiation game over* G, denoted as $N_d(G)$, is a game over $d + 1$ rounds which are labeled as: $t_0, t_1, t_3...t_d$. The first d rounds are referred to as the *negotiation stage*, and the last round is called the *action stage*. The idea is that only in the last round the players take an action from the game G, while during the first d rounds the players negotiate which actions they will take in the last round.

We will now define the negotiation protocol that is applied during the negotiation stage. In each round of the negotiation stage each player takes an action which is either: 'accept(x)', or 'none', where x can be any deal from the agreement space Agr_G associated with the game G. The players take their actions simultaneously, and the accepted deal x can be different for each of the players.

Definition 6 We say a deal x is **confirmed in round** t_j if j is the smallest number for which both the following predicates are true:

- For each $\alpha_i \in pa(x)$ there is some round t_{k_i} with $k_i \leq j$ in which α_i has taken the action 'accept(x)'.
- The set $X_j \cup \{x\}$, where X_j is the set of all deals that have been confirmed in any round t_k with $k < j$, is consistent.

We say a deal x is **confirmed** if it was confirmed in any round t_j with $j < d$.

This means that a deal x is considered confirmed if at some point all its participating agents have played 'accept(x)', and x is consistent with the deals that have already been confirmed earlier.

This negotiation protocol is called the Unstructured Negotiation Protocol, and was introduced in [4]. It allows each player to make any proposal whenever he or she wants, unlike the more common Alternating Offers Protocol [29] in which players take turns. The definition of a negotiation game could be easily changed to reflect the Alternating Offers Protocol instead, but we think that this protocol is too restrictive to model real negotiations. In fact, the rules of Diplomacy do not specify any protocol at all. Players are allowed to negotiate however they want.

In most literature on negotiation protocols one agent *proposes* a deal, and then another agent may or may not *accept* the deal. To keep our formalization simple however we do not make this distinction, so both the proposal and the acceptance are represented by the 'accept' action. Furthermore, we should note that under this protocol the negotiators are not obliged to respond to a proposal. Instead of rejecting it or making a counter proposal they may simply remain silent, by playing the 'none' action. Furthermore, we do not explicitly model the option of withdrawing from the negotiations. However, a negotiator can still withdraw simply by remaining silent for the rest of the negotiation stage.

If X_d denotes the set of all deals that were confirmed during the negotiation stage, then during the action stage of $N_d(G)$ the players play the game $G[X_d]$. This means they can only pick actions that are consistent with the agreements they made during the negotiation stage.

Definition 7 Let G be a one-shot game and d a positive integer. The **Negotiation Game over** G, denoted as $N_d(G)$, is a game over $d+1$ rounds, with the same players as G. In the first d rounds (**the negotiation stage**) in each round each player can play either the action 'none' or an action accept(x) for any $x \in Agr_G$, and in the last round (**the action stage**) the players play the game $G[X_d]$, where X_d is the set of deals confirmed during the negotiation stage. Each player α_i receives the utility $f_i^G(\mu)$ where $\mu \in \bigcap_{x \in X_d} x$ is the action profile chosen by the players during the action stage.

For simplicity we have here defined the negotiations to take place over a sequence of d discrete rounds. However, one can easily use this model to approximate negotiations that take place in continuous time, simply by taking d to be a very large number and setting the duration of each round in the negotiation stage to a very small number. This may mean that players do not have enough time to decide which action to take in each round, but that is not a problem if we assume that not taking an action is interpreted as taking the action 'none'. In the following, we will often write $N(G)$ instead of $N_d(G)$ since the actual value of d is mostly irrelevant for our purposes.

Finally, we would like to stress that a player's set of allowed actions $\mathcal{M}_i^G[X_d]$ in the action stage can only be smaller than its full set of actions $\mathcal{M}_i^G$ if that player α_i himself has agreed with those restrictions by accepting the deals in X_d. Of course, restricting your set of actions is never beneficial by itself, but when the other players return the favor by also restricting their sets of actions, this can be highly beneficial.

Example Again, let G be the Prisoner's Dilemma, then $N_1(G)$ is the negotiation game over the Prisoner's Dilemma with $d = 1$. In round t_0 both players have the opportunity to suggest a deal; for example, they could play the action accept($\{deny\} \times \{\text{deny}\}$). If they both suggest this deal then this deal is confirmed, meaning that in round t_1 each player α_i can only play the action 'deny'.

Proposition 1 *If G is the Prisoner's Dilemma then the negotiation game $N_1(G)$ has a Subgame Perfect Equilibrium that consists of both players playing accept($\{deny\} \times \{deny\}$) in t_0 and both playing 'deny' in t_1.*

Note that the Nash Equilibrium of $N_1(G)$ dominates the Nash Equilibrium of the pure Prisoner's Dilemma G without negotiations, in which both players play 'confess'. Therefore, this demonstrates how the introduction of negotiations improves the individual outcomes of the players.

In the introduction we mentioned that we want utility functions to satisfy three properties. We see that for any Negotiation Game the second and third property are indeed satisfied: the utility values obtained by the players are determined by the actions they take in the action stage, rather than the agreements made in the negotiation stage. The agreements only influence the players' utilities indirectly, because they restrict the actions that can be taken during the action stage. We think this model reflects how negotiations work in real life. After all, signing a contract is not an action that is valuable by itself. Instead, it merely binds all signing parties to undertake certain actions (or refrain from undertaking certain actions), and it is these actions that determine the utilities obtained by those parties.

5 D-Brane

We have implemented a Diplomacy-playing agent called D-Brane (Diplomacy BRAnch & bound NEgotiator). As a heuristic, in each round it tries to maximize the number of Supply Centers conquered during that round. In this way we can regard the action stage of a single round of Diplomacy as a game in itself, which we denote as Dip_ϵ. The parameter ϵ represents the configuration of the units on the map, which is of course different in each round. Our agent's utility function f for the game Dip_ϵ is then the number of Supply Centers conquered, and a full round of Diplomacy can then be seen as an instance of the negotiation game $N(Dip_\epsilon)$. A formal definition of Dip_ϵ can be found in Appendix B.

Of course, the fact that our player only tries to maximize its number of Supply Centers for the current round is a very greedy heuristic. We know from personal experience that real players often think ahead more than one round. Nevertheless, as we will see in the experimental section, we did manage to implement a successful player using this heuristic.

D-Brane consists of two independent components: the strategic component (see Section 5.1) and the negotiating component (Section 5.3). The negotiating component searches for deals during the negotiation stage to propose to the coalition partners and determines whether or not to accept proposals received from the coalition partners. The strategic component determines, given any deal x, which are the best actions to take if x is confirmed. The strategic component is used in two ways: during the negotiation stage the negotiating component applies the strategic component to determine whether any deal is valuable enough to be proposed or accepted, and during the the action stage it is used to determine which orders to submit, under the restriction of the agreements that were made during the negotiation stage. This modular decomposition of D-Brane is important because it will allow future researchers to replace D-Brane's negotiation algorithm with their own algorithms, allowing them to compare these negotiation algorithms independent of the underlying strategy.

It is important to understand that D-Brane is a selfish negotiator: it does help its allies in the game, but only because it expects help from them in return in later rounds. Another important aspect of D-Brane is that it always obeys all agreements it makes and always assumes that the other agents will do the same. This is a simplification, because in a real Diplomacy game players may not always obey their agreements, and therefore the notion of trust is an important factor. However, the subject of trust is beyond the scope of our work so we only focus on strategy and negotiations. Furthermore, D-Brane does not try to form the best possible coalition. Instead, it assumes that a certain coalition structure is given from the start. Again, this is because the problem of coalition formation is beyond the scope of our work.

D-Brane was implemented in Java, using the DipGame framework [9]. However, we did not use the negotiation language and negotiation server provided by DipGame, because it turned out easier to implement our experiments with a custom made negotiation server and language, which we have bundled into the new BANDANA framework (see Appendix A).

In the following, we will always assume that the algorithms we describe are running on the agent with name α_1. The other agents, $\alpha_2 \dots \alpha_n$, may also be copies of D-Brane, but they may just as well be any other Diplomacy playing agent, or they may even be human players.

5.1 The strategic component

Given a game state ϵ (i.e. a configuration of units on the Diplomacy map), a deal x and any player α_i the strategic component returns a set of orders for α_i for the game $Dip_\epsilon[x]$ plus the number of Supply Centers that α_i is guaranteed to conquer if it submits those orders. Note that, although this component is part of agent α_1, it can also be used to predict the orders and the the number of conquered Supply Centers of any other player α_i. This is important, because it allows α_1 to assess how valuable the deal x is to the other participants in the deal. After all, it does not make sense to propose deals that are unprofitable for the other participants.

In theory one could try to determine the set of all possible actions for each player and then calculate the Nash Equilibrium, but this is computationally too expensive as the number of possible action profiles can easily be of the order 10^{10}. Our algorithm, however, manages to quickly make very good approximations by decomposing the game into a number of smaller games which each correspond to the battle for a single Supply Center. Then, after determining the best battle plans for each battle, it tries to find the strongest combination of such battle plans.

We say that a battle plan for a player α_i to conquer a province p is *invincible* if no opponent can choose any battle plan that would prevent α_i from conquering p (see Appendix B for a formal definition). Once we have determined all battle plans for a given province p, for all players, it is easy to determine which of those battle plans are invincible. Similarly, we can determine the *invincible pairs* of battle plans. An invincible pair is a pair of battle plans (β_p, β_q) for two different Supply Centers p and q

respectively, such that if α_1 plays both of these battle plans, then at least one of them is guaranteed to succeed (again, see Appendix B for a formal definition). The idea behind the definition of an invincible pair is the following. Suppose that our battle plan β_p can be defeated by an opponent's battle plan β'_p and that our battle plan β_q can be defeated by an opponent's battle plan β'_q. One might be inclined to jump to the conclusion that playing β_p and β_q cannot guarantee us to conquer any Supply Center. However, it might be the case that β'_p and β'_q are the *only* plans that can defeat β_p and β_q, and that β'_p and β'_q are incompatible with each other, so the opponent cannot play them both. In that case, if we play both β_p and β_q we are still guaranteed that at least one of the two will succeed.

5.1.1 The basic algorithm

If ϵ denotes some state of the game the set of all battle plans for player α_i to conquer or defend a Supply Center p is denoted $B^{\epsilon}_{i,p}$. The set of all Supply Centers is denoted SC.

Given a game state ϵ, an agreement x, and a player α_i, the strategic component works as follows:

1. For each $p \in SC$ determine all invincible plans from the set $B^{\epsilon}_{i,p}$.
2. For all pairs of Supply Centers $(p, q) \in SC \times SC$ (with $p \neq q$) determine all invincible pairs from the set $B^{\epsilon}_{i,p} \times B^{\epsilon}_{i,q}$.
3. Remove all invincible plans and invincible pairs that are not consistent with x.
4. Find the largest consistent combination of invincible plans and pairs, using And/Or tree search with Branch & Bound (Section 5.1.2).
5. For each province for which we have not been able to select an invincible plan or pair, select the strongest non-invincible battle plan that is consistent with x and the plans and pairs chosen in the previous step.
6. Return the full set of battle plans we have selected.

The number of invincible plans and pairs returned equals the number of Supply Centers α_i is guaranteed to conquer. The plans chosen in step 5 merely form a best effort to try to conquer some more Supply Centers even though the opponents might be able to defeat those attempts. Of course, the opponents may not be perfect so it is at least worth trying.

The battle plans in $B^{\epsilon}_{i,p}$ may contain support orders for units of coalition partners of α_i. However, α_i cannot be sure that these coalition partners will indeed submit those orders, unless the deal x ensures that. Therefore, any plan that contains orders for units that do not belong to α_i and that are not ensured by the agreement x are discarded.

In theory, the algorithm would be even stronger if it did not only determine invincible plans and pairs, but also invincible triples, invincible quadruples, etcetera. However, the number of such n-tuples grows exponentially with n, so checking which ones are invincible would slow down the algorithm considerably.

5.1.2 And/Or tree search

And/Or tree search [5] is a variant of regular tree search that can be used to solve Constraint Optimization Problems. The power of this technique lies in the fact that the depth of the search tree is drastically decreased with respect to a naive tree search, by exploiting the knowledge that certain variables are independent.

Note that step 4 of the algorithm above is indeed a Constraint Optimization Problem: for each Supply Center we want to pick an invincible plan or an invincible pair, but not all combinations of such plans and pairs are consistent. Since every invincible plan or invincible pair guarantees a conquered Supply Center, we aim to pick as many of such plans as possible. However, because of the constraints we may not be able to pick an invincible plan or pair for every Supply Center; we sometimes need to pick the 'empty plan' (i.e. none of our units will try to conquer the Supply Center).

The variables of this constraint optimization problem are the Supply Centers $p \in SC$. For each such variable, its domain consists of the set of invincible plans, the set of invincible pairs,[6] and the empty plan. The constraints between the variables are given by the fact that a battle plan for Supply Center p and a battle plan for Supply Center q may be incompatible if they contain two different orders for the same unit. The value to optimize is the number of Supply Centers for which we have not chosen the empty plan.

In this setting it is very easy to see which variables depend on each other, and which are independent. Two Supply Centers p and q are independent if there is no unit that is involved in the battle plan for p as well as in a battle plan for q. This means that it is an ideal case for And/Or Tree Search.

5.2 Constraint optimization games

Above we have shown how our player solves the complex game Dip_{ϵ} by decomposing it into smaller games (the battles), determining the best moves in every such battle, and then using Constraint Optimization techniques to find the best combination of these best moves.

However, the way we presented this was completely ad-hoc for the game of Diplomacy. In this section we will therefore generalize these ideas, by only looking at the abstract properties of the game that allowed us to follow

[6]An invincible pair for provinces (p, q) is considered a value for the variable corresponding to p, where p is lower than q in some predefined ordering of the Supply Centers.

this 'divide-and-conquer' approach. We will define a new class of games that we call Constraint Optimization Games (COG). A COG is a game that can be decomposed as a number of smaller games that are played simultaneously but that are not independent. We then show how the above described algorithm for Diplomacy can be generalized to any other Constraint Optimization Game.

Let MG be a set of m one-shot games: $MG = \{G_1, G_2, ...G_m\}$, each with the same n players. We call these games the **micro-games**. The set of actions for player α_i in micro-game G_j is denoted as $\mathcal{M}_i^{G_j}$.

Definition 8 (**A Constraint Optimization Game**) G_{MG} is a game with n players, such that the set of actions $\mathcal{M}_i^{G_{MG}}$ for player α_i in G_{MG} is a subset of the Cartesian product of the actions of each micro-game: $\mathcal{M}_i^{G_{MG}} \subseteq \mathcal{M}_i^{G_1} \times \mathcal{M}_i^{G_2} \times ...\mathcal{M}_i^{G_m}$. The utility function for a player α_i is defined as the sum of its utility functions in each of the micro-games:

$$f_i^{G_{MG}}(\mu) = \sum_{j=1}^{m} f_i^{G_j}(\mu_j) \qquad (1)$$

Note that an action profile μ of G_{MG} can then be viewed as a matrix in which each row μ_j represents an action profile from the micro-game G_j and each entry $\mu_{j,i}$ represents the action taken by player α_i in micro-game G_j.

We see that the game G_{MG} consists of a number of smaller one-shot games that cannot be played independently from each other, because the set of actions $\mathcal{M}_i^{G_{MG}}$ of α_i is a *subset* of the product of the sets $\mathcal{M}_i^{G_j}$. Each player α_i can pick for any micro-game G_j any action from $\mathcal{M}_i^{G_j}$, but not all combinations of such actions are allowed. In other words: there are constraints between the actions of the several micro-games that need to be satisfied. Therefore, the best strategy for the game G_{MG} is not simply the combination of the best strategies of each individual micro-game. For example, player α_1 may have the optimal action $a \in \mathcal{M}_i^{G_1}$ in micro-game G_1 and an optimal action $b \in \mathcal{M}_i^{G_2}$ in micro-game G_2, but the combination of a and b may be illegal, so α_1 is forced to choose a suboptimal action in at least one of these two micro-games.

Example In the game Dip_ϵ the battle for a single Supply Center p_j can be seen as a micro-game G_j. The utility for a player α_i in micro-game G_j is 1 if α_i succeeds in conquering p_j, and 0 otherwise. Its utility for the entire COG is then the total number of Supply Centers it conquers. For each Supply Center p a player α_i must choose which orders to submit for its units located around p. However, some of those units may also be adjacent to another Supply Center q. Therefore, if α_i decides to order a unit u to move to p, it can no longer use that unit to move to q. So we see that indeed there are constraints between the micro-games.

The concept of a COG combines aspects from Constraint Optimization Problems (COP) with aspects from Game Theory. Just like in a COP an agent α_i needs to pick m values for m different variables, such that the combination is consistent and maximizes its utility [30]. However, unlike normal COPs, the utility of an agent does not only depend on the values it chooses, but also on those chosen by its opponents, which have different utility functions to maximize, just as in Game Theory. Another way to look at it, is to see it as a variation of a Distributed Constraint Optimization Problem in which there is not one single utility function to maximize, but rather each agent aims to maximize its own individual utility function.

Let us now present a rough sketch of how one can generalize the algorithm described in Section 5.1 to other COGs. The essence of our algorithm can be described in three steps:

1. Assign a value to every action of α_i in every micro-game.
2. Do the same for every pair of actions of α_i for every pair of micro-games.
3. Use And/Or search to find the combination of actions that maximizes the sum of its action-values, under the restriction that the chosen set of actions must be consistent.

The idea is that assigning a single value to each action in each micro-game in fact reduces the COG to a standard COP, which can be solved with an And/Or search. One straightforward way to assign a value to an action $\mu_{j,i}$ is to find the set of opponent actions $\mu_{j,-i}$ that minimize the utility function $f_i(\mu_{j,i}, \mu_{j,-i})$. In other words: the value obtained in the worst-case scenario that the opponents pick those actions that minimize α_i's utility. If we apply this principle to Dip_ϵ, then this means that every invincible plan or pair receives a value of 1 and all other plans receive a value of 0, which essentially means that all non-invincible plans are discarded, which is indeed what we did.

Finally, we would like to remark that finding the best actions to take in a COG is a hard combinatorial problem, so if G is a COG, then $N(G)$ satisfies all three properties that we mentioned in the introduction.

5.3 The negotiating component

During the negotiation stage of $N(Dip_\epsilon)$, given a coalition $C \subset Pl$ that includes α_1, the negotiating component explores the agreement space by means of a best-first Branch & Bound tree search in order to find good deals to propose to the coalition partners. At regular time intervals it determines whether it should make a new proposal to its coalition partners and, if yes, which one. Furthermore, whenever the agent receives a proposal from any of the coalition partners it determines whether to accept that proposal or not.

The negotiating tree search algorithm is a previously developed algorithm for general negotiation settings, called NB^3, applied to Diplomacy. We only give a brief description here. For more information we refer to [4].

5.3.1 Tree search

We will now use the notation $p_1, p_2, \dots p_{34}$ to denote the 34 Supply Centers of Diplomacy. Let C denote the set of coalition partners of α_1 (including α_1 itself), let $B^{\epsilon}_{C,p}$ denote the set of battle plans with target $p \in SC$ involving only players in C, and let B^{ϵ}_{C} denote the set of battle plans for all coalitions partners, for any Supply Center:

$$B^{\epsilon}_{C} = \bigcup_{p \in SC} B^{\epsilon}_{C,p} \qquad B^{\epsilon}_{C,p} = \bigcup_{\alpha_i \in C} B^{\epsilon}_{i,p}$$

The NB^3 algorithm expands a search tree in which each node ν (except the root node) is labeled with a battle plan $\beta_\nu \in B^{\epsilon}_{C,p}$ for some Supply Center p. The search starts with a single tree node (the root). Then, if B^{ϵ}_{C,p_1} has size k_1, the algorithm will add k_1 child nodes to the root, with each child labeled with a different battle plan $\beta \in B^{\epsilon}_{C,p_1}$. Next, it uses a heuristic function to determine which of these new nodes is the "best" and continues expanding that node ν. If B^{ϵ}_{C,p_2} has size k_2 then the algorithm creates k_2 children for node ν, each labeled with a different battle plan $\beta \in B^{\epsilon}_{C,p_2}$. Again, the heuristic function chooses the next node to split, and this is repeated until either the deadline for negotiations has finished, or until the tree has been explored exhaustively. Any node for which the battle plan is incompatible with the battle plans of its ancestors is pruned immediately.

Each node in this tree represents a deal that may be proposed to the coalition partners. Specifically, if $path(\nu)$ denotes the set of nodes that make up the path from the root node to ν, and $plan(\nu)$ the set of battle plans corresponding to the labels of the nodes in $path(\nu)$:

$$plan(\nu) = \{\beta \in B^{\epsilon}_{C} \mid \exists \nu' \in path(\nu) : \beta = \beta_{\nu'}\}$$

then with each such plan $plan(\nu)$ we can associate a deal $x_\nu \in Agr_{Dip_\epsilon}$ in which each participating player is committed to submit his or her orders[7] that appear in $plan(\nu)$, which means that any action profile μ is allowed, as long as it contains all orders that the players have committed themselves to.

Note that x_ν as defined here is indeed a deal in the sense of Definition 2. The battle plans in $plan(\nu)$ consist of a number of orders, and if for some player α_i there are a number of orders in $plan(\nu)$ then its restricted set of actions $\mathcal{M}^{\epsilon}_{i}[x_\nu]$ consists of those actions in $\mathcal{M}^{\epsilon}_{i}$ that contain all α_i's orders in $plan(\nu)$.

Note that the set of deals that are considered in this way is much smaller than the full agreement space Agr_{Dip_ϵ}, because we are only looking at deals in which players commit themselves to battle plans, rather than any random set of orders. In this way we have filtered out for example actions containing invalid supports, and action profiles in which coalition partners take contradictory actions (e.g. two coalition partners both trying to attack the same province).

In order to determine which of the deals represented in the tree are good enough to propose to the coalition partners our algorithm calculates for each node ν and each coalition partner $\alpha_i \in C$ a utility value $u_i(x_\nu)$. This utility value is not the same as the utility function f as defined for Dip_ϵ, but rather it is a value that indicates how profitable the deal x_ν is to player α_i. In Section 5.3.2 we will explain how $u_i(x_\nu)$ is defined.

Furthermore, for each node ν and for each coalition partner $\alpha_i \in C$ the algorithm stores an upper bound $ub_i(\nu)$ and a lower bound $lb_i(\nu)$, which are used for pruning and to calculate the search heuristic. The upper bound $ub_i(\nu)$ is the highest utility $u_i(x_{\nu'})$ agent α_i could possibly receive from any plan $x_{\nu'}$ where node ν' is any descendent of ν. This means that if $ub_i(\nu)$ is lower than α_i's reservation value, any plan that could appear in the subtree under ν would be unprofitable for α_i so the node ν can be pruned. Similarly, the $lb_i(\nu)$ is the lowest possible value of $u_i(x'_\nu)$ then one could find for any node ν' that is a descendant of ν.

If for a certain node ν the utility value $u_i(x_\nu)$ is higher than the reservation value rv_i for every participating agent $\alpha_i \in pa(x_\nu)$, it means the deal is in principle profitable for every participating agent, and therefore D-Brane may consider to propose it to the others. In that case, the deal will be stored in a list of potential proposals, which will be used by the negotiation strategy as we will explain in Section 5.3.3.

5.3.2 Credits

The algorithm tries to find deals that are profitable for each player participating in the deal. For example, α_1 may support α_2 to attack some Supply Center p while in return α_2 will support α_1 to attack some other Supply Center q. This is mutually beneficial if neither of these players is able to conquer its targeted Supply Center without support from the other player. Unfortunately, it turns out that such situations in which two or more players have the opportunity to help each other do not occur very often.

To increase the number of opportunities to make beneficial deals D-Brane applies a 'credit' system, which means that when one player α_i gives support to another player α_j,

[7]Remember that $plan(\nu)$ is a set of battle plans, and each battle plan is a set of orders. So with "the orders in $plan(\nu)$" we actually mean the orders in the battle plans in $plan(\nu)$.

the supported player α_j is considered indebted to α_i. This means that α_i can expect α_j to return the favor and give support to α_i at some later phase of the game. Thanks to this system a player may be willing to support another player even if the favor cannot be returned immediately, which strongly increases the number of potential deals.

In order to assess the value of a deal, a player should therefore not only take the number Supply Centers that are conquered thanks to the deal into account, but also assign some extra value to the deal to represent future supports that may be received as a consequence of the current deal.

Specifically, D-Brane stores a number $d_{i,j}$ which is the *credit balance* between players α_i and α_j: the number of supports α_i has so far given to α_j, minus the number of supports α_j has given to α_i. If $d_{i,j} < 0$ it means that α_j still owes a number of supports to α_i.

D-Brane then calculates the value $u_i(x)$ of a deal x for player α_i as:

$$u_i(x) = E(f_i^{Dip_\epsilon[x]}) - E(f_i^{Dip_\epsilon}) + \sum_{j=1}^{n} Cr(d_{i,j}) \tag{2}$$

where $E(f_i^{Dip_\epsilon[x]})$ is the expected number of conquered Supply Centers in the current turn given the deal x, where $E(f_i^{Dip_\epsilon})$ is the expected number of conquered Supply Centers in the current turn if no deal is made, and $Cr(d_{i,j})$ is given by:

$$Cr(d_{i,j}) = \begin{cases} 0.4 \cdot d_{i,j} & \text{if } d_{i,j} \geq 0 \\ 0.6 \cdot d_{i,j} & \text{if } d_{i,j} \leq 0 \end{cases} \tag{3}$$

One should understand that while $f_i^{Dip_\epsilon[x]}$ is the short-term utility (the number of Supply Centers conquered directly in the current round), the utility $u_i(x)$ is a sort of long-term expected utility value, which takes into account the number of supports α_i still owes to other players and the number of supports other players still owe to α_i.

We will now show that the values 0.4 and 0.6 that appear in (3) are chosen such that D-Brane exhibits behavior that one would indeed expect from a selfish negotiator.

Proposition 2 *A rational player α_i that evaluates deals according to (2) would only be willing to give support to another player α_j if that does not cause α_i to lose a Supply Center.*

Proof Losing a Supply Center causes $E(f_i^{Dip_\epsilon[x]})$ to decrease by 1, while giving support only increases $Cr(d_{i,j})$ by either 0.4 or 0.6, so in total u would decrease. □

Proposition 3 *A rational player α_i that evaluates deals according to (2) would only ask support from another player α_j if α_i expects that this will yield an extra Supply Center for α_i.*

Proof If no extra Supply Center is gained the received support decreases u by either 0.4 or 0.6. □

Proposition 4 *If a rational player α_i that evaluates deals according to (2) has a positive credit balance w.r.t α_j, then α_i would prefer to receive support from α_j and gain a Supply Center, rather than give more support to α_j.*

Proof If α_i has a positive credit balance, then the combination of a gained Supply Center and a received support increases u_i by $1 - 0.4 = 0.6$, while giving more support would increase u_i by only 0.4. □

Proposition 5 *If a rational player α_i that evaluates deals according to (2) has a negative credit balance w.r.t α_j, then α_i would prefer to give support to α_j rather than to ask for more support from α_j and gain a Supply Center.*

Proof With a negative balance, the extra support would yield $1 - 0.6 = 0.4$, while giving support would increase u_i by 0.6. □

Without Proposition 2 D-Brane would not be selfish because it would be inclined to constantly give support to others, while losing its own Supply Centers. Proposition 3 guarantees that D-Brane does not request any unnecessary supports. Proposition 4 makes sure that D-Brane only gives support because it expects that others will return the favor. And without Proposition 5 D-Brane would never be willing to return any favors, and hence nobody would like to negotiate with D-Brane.

5.3.3 Negotiation strategy

Every time a new node ν is generated the algorithm determines whether the corresponding deal x_ν is rational to all participating agents $pa(x_\nu)$. That is, it checks whether for each participating agent the utility $u_i(x_\nu)$ is larger than its reservation value rv_i. If this is the case, then it is stored in a repository of potential deals.

Then, at set intervals during the negotiation stage, the NB^3 negotiation algorithm determines whether any of the deals in this repository can be proposed. It does this by applying a time-based strategy: the closer to the deadline, the more it is willing to propose or accept deals with less utility for α_1. Furthermore, the closer to the deadline, the more it will be inclined to propose deals that yield high utility for the other participating agents.

5.3.4 Implicit vs. explicit agreements

When two or more players in Diplomacy are in a coalition, this usually implies two things: it means they will not attack each other and it means that they may support each other when attacking an opponent. We call the first kind of agreement an **implicit agreement** because it is implied by the fact that the players are allies, so no negotiation is needed to establish such agreements. The second kind we call an **explicit agreement** and can only be made by negotiating. The deals investigated and proposed by the Negotiating Component are therefore exclusively of the explicit type. The ability of D-Brane to negotiate (i.e. to make explicit agreements) and its ability to obey implicit agreements can both be switched on and off, so it can play in 4 different modes: with negotiations on or off and with implicit agreements on or off.

It is important to make this distinction for the experimental evaluation of D-Brane. On the one hand it is unrealistic to play without implicit agreements, because any rational and trustworthy player would always apply them. On the other hand however, we want to investigate the importance of negotiations. Therefore, if D-Brane defeats its opponents, we want to know whether this is caused by its negotiation algorithm, or by the fact that it has a strong strategic module, or simply because it plays in a coalition and the mere fact that the coalition partners do not attack one another gives them an advantage over the opponents. Testing the agent in all 4 different modes allows us to identify to what extent each of these three elements is responsible for playing well.

Let X_d denote the set of explicit agreements that were confirmed during the negotiation stage of $N(Dip_\epsilon)$, and let X_C denote the set of implicit agreements implied by the coalition C that α_1 is in. Then during the action stage of $N(Dip_\epsilon)$ the strategic component tries to determine the best possible action of the game:

- $Dip_\epsilon[X_d \cup X_C]$, if both implicit and explicit agreements are obeyed.
- $Dip_\epsilon[X_d]$, if only explicit agreements are obeyed.
- $Dip_\epsilon[X_C]$, if only implicit agreements are obeyed.
- Dip_ϵ, if no agreements are obeyed.

In order to correctly interpret the experimental results we feel it is important to stress that *if both implicit and explicit agreements are switched off this essentially means that D-Brane plays entirely individually, and does not consider itself part of any coalition.*

6 Experiments

In this section we compare D-Brane with two other Diplomacy playing agents (also known as 'bots') recently presented in [8, 13]. In both papers a negotiating agent was tested by letting it play against a standard non-negotiating bot called the DumbBot. We have done similar experiments with our D-Brane agent and compared the results with theirs. Moreover, we have submitted D-Brane to participate in the Computer Diplomacy Challenge. Our agent turned out to be the winner of this competition, which confirms the results of our experiments.

We did experiments with the number of D-Branes varying from 2 to 5, while all other players always being DumbBots. Every such experiment was repeated 4 times; one time for each different mode of D-Brane. In each of the experiments we performed with negotiations off (i.e. not making any explicit agreements) we played 500 games, which took up to four-and-a-half hours. For experiments with negotiations we set the deadline for negotiations to 5 s per round. Since such experiments took much more time we only played 200 games in each such experiment, which still took up to 17 h. In order to prevent games from continuing forever because they get stuck in a stalemate, we have programmed the agents to automatically declare a draw after 40 rounds.[8] For our experiments we used the Parlance game server.[9] In each new game this server randomly determines which player will play which Great Power.

In all experiments with implicit agreements turned on, the D-Branes were instructed to form a coalition against the DumbBots. D-Brane never breaks any promises and never leaves the coalition, and assumes its coalition partners will not do so either. As explained, this is because trust and coalition formation are beyond the scope of our work. All experiments were performed on a single HP Z1 G2 desktop computer with Intel Xeon E3 4x3.3GHz CPU and 8 GB RAM.

6.1 D-Brane vs. DumbBot

As explained above, we did a number of experiments, with the number of D-Branes in each experiment varying between 2 and 5, and for each of these numbers, we performed an experiment with each of the 4 modes of D-Branes, resulting in a total of 16 experiments. For each of these experiments we measured the performance of D-Brane by counting the number of Supply Centers conquered. In total there are 34 Supply Centers, and 7 players, so if all players are equally strong we can expect each player to conquer $\frac{34}{7}$ Supply Centers, so if there are n D-Branes then we can conclude that D-Brane is better than DumbBot if the D-Branes obtain more than $n \cdot \frac{34}{7}$ Supply Centers. We see in

[8]For readers more familiar with Diplomacy: we mean that the players declare a draw whenever a game reaches the Winter 1920 phase.

[9]https://pypi.python.org/pypi/Parlance/1.4.1

Table 1 that this was clearly the case in every experiment. We can therefore conclude that *D-Brane plays significantly better than DumbBot*. For example, in the case of 4 D-Branes and 3 DumbBots, without negotiations and without implicit agreements, on average the D-Branes together conquer almost 30 Supply Centers, leaving only 4 Supply Centers for the DumbBots.

As expected, we see that playing with implicit agreements improves the outcome for the D-Branes. However, we also see that playing with negotiations only improves the result if it is in combination with implicit agreements. Surprisingly, if seems that if the implicit agreements are turned off, the negotiation algorithm even has a detrimental effect, although this is effect too small compared the standard errors to call it significant.

It is at this point unknown to us why exactly negotiations seem to be ineffective without implicit agreements. One hypothesis is that this effect might be caused by the fact that without implicit agreements but with negotiations the D-Branes will mainly spend their efforts on closing deals to protect themselves from one another, and can therefore spend less effort on eliminating DumbBots. Without explicit agreements and without negotiations, it would be easier for one D-Brane to quickly eat up another D-Brane, which would lead to one very strong D-Brane, which could then focus on eliminating the DumbBots. Of course, this is bad for the eliminated D-Branes, but for the group of D-Branes as a whole this could lead to a higher total number of conquered Supply Centers. With implicit agreements and with negotiations, the D-Branes do not attack each other anyway, so all their efforts can be focused entirely on eliminating the DumbBots. More research is required to determine whether this hypothesis holds or not.

Table 1 Number of Supply Centers (± standard error) conquered by the D-Branes in various settings

	nego off	nego on
2xD-Brane vs. 5xDumbBot		
impl. agr. off	15.5 ± 0.3	15.1 ± 0.5
impl. agr. on	17.2 ± 0.3	17.9 ± 0.5
3xD-Brane vs. 4xDumbBot		
impl. agr. off	24.9 ± 0.3	24.4 ± 0.5
impl. agr. on	28.4 ± 0.2	29.6 ± 0.4
4xD-Brane vs. 3xDumbBot		
impl. agr. off	29.8 ± 0.2	29.5 ± 0.3
impl. agr. on	31.5 ± 0.1	32.1 ± 0.2
5xD-Brane vs. 2xDumbBot		
impl. agr. off	32.0 ± 0.1	31.5 ± 0.2
impl. agr. on	32.3 ± 0.1	33.2 ± 0.1

There are 34 Supply Centers in total

Table 2 The average rank obtained by the D-Branes (± standard error)

2xD-Brane vs 5xDumbBot		
	nego off	nego on
impl. agr. off	2.59 ± 0.05	2.75 ± 0.08
impl. agr. on	2.42 ± 0.05	2.35 ± 0.08

The theoretically optimal value is 1.5, and the theoretically worst is 6.5. A value of 4.0 would mean D-Brane is equal to DumbBot. DipBlue achieves 3.57

6.2 D-Brane compared with DipBlue

In [13] a negotiating Diplomacy agent called DipBlue was introduced. In their experiments they let 2 instances of DipBlue play against 5 DumbBots. As a measure of success they used the average rank of their two agents over 75 games. That is: after each game the best player gets rank 1, the second best player gets rank 2, etcetera, and the worst player gets rank 7. A player is considered better than another player if it finishes with more Supply Centers, or if it is eliminated later. If all players in a game are equally strong then you may expect all players to receive an average rank of 4.0. With 2 instances of the player to test, the best possible average rank is 1.5 and the worst is 6.5. The best average rank that DipBlue achieves in their experiments[10] is 3.57.

To compare the performance of D-Brane with DipBlue we have also measured the average rank of D-Brane in the experiments with 2 instances of D-Brane against 5 instances of DumbBot. The results are displayed in Table 2. It displays the average rank of the D-Branes in the four different modes, with their respective standard errors. We note that in all cases the average rank is around 2.5, even when negotiations were switched off. This means that not only our agent is better than the DumbBot, but also that *even without negotiating D-Brane plays significantly better than DipBlue with negotiations*.

We should remark here, that comparing the negotiation skills of D-Brane with the negotiation skills of DipBlue is not entirely fair, because D-Brane always obeys all confirmed agreements, and assumes its coalition partners obey all confirmed agreements, whereas DipBlue takes into account that the other players may not obey their commitments. Thus, when negotiating, D-Brane has an unfair advantage with respect to DipBlue. However, this advantage is not present when both implicit and explicit agreements are

[10]To be precise: this result was obtained in case one DipBlue was negotiating with one instance of slightly more simplistic agent called ‘Naive’ against five DumbBots. In case two instances of DipBlue were playing, their result was worse than 3.57.

turned off, since in that case D-Brane essentially plays with no coalition partners at all. So at least for that case it is fair to say that D-Brane outplays DipBlue. The same also holds for our statement that D-Brane outplays the DumbBot.

6.3 D-Brane compared with Fabregues' agent

In [8] a nameless agent was presented by Fabregues, which was also compared with the DumbBot. Fabregues did 8 experiments, in which the number of instances of her bot varied between 0 and 7, and the remaining players were DumbBots. In each experiment she played 100 games. As a measure of success she counted the number of victories of her agent. To compare our results with hers, we have also counted the number of victories of D-Brane in our experiments and displayed them in Table 3.

It is important to note here, that Fabregues' experiments were performed on a super computer, with deadlines of 5 minutes per round, while we did our experiments on a desktop computer, and with deadlines of only 5 s per round. Furthermore, the results displayed in the table are for the cases that the D-Branes did not negotiate at all, and played without implicit agreements.

Here, we count a 'victory' if a D-Brane ends with the highest number of supply centers. If 2 agents both end with the highest supply centers we count 'half a victory' for each of them. We should note that we have rounded off the percentages displayed for Fabregues' agent to multiples of 5. This is because in her thesis Fabregues does not provide the exact percentages, but only displays a graph from which it is hard to read off the exact values.

We see that in all cases *D-Brane scored significantly better than Fabregues' negotiating agent, even if D-Brane was not negotiating, and even though our experiment ran on a single desktop computer rather than on a super computer, and even though our deadlines were much shorter*.

6.4 Varying negotiation deadlines

In order to see how the length of the deadlines affects the performance of our negotiation algorithm we have done some more experiments, with varying deadlines. In these cases the we had 3 D-Branes playing against 4 DumbBots,

Table 3 First row shows the victory percentages of n instances of Fabregues' agent versus $7 - n$ DumbBots

n	2	3	4	5
Fabregues' agent	45 %	70 %	85 %	95 %
D-Brane	61%	84 %	95 %	98 %

Second row shows the same, for n D-Branes versus $7 - n$ DumbBots

Table 4 Number of Supply Centers conquered by the D-Branes, with varying negotiation deadlines

	3xD-Brane vs. 4xDumbBot			
Deadline per round	5 s	10 s	15 s	20 s
SC's conquered	29.6 ± 0.4	29.9 ± 0.8	29.9 ± 0.7	29.5 ± 0.8

with negotiations and with implicit agreements. The results are displayed in Table 4. Each of the results is an average over 50 games, except in the case of 5 s, which was averaged over 200 games. We conclude from this experiment that increasing the negotiation time does not improve the results any further. If D-Brane is capable of finding a potential deal, it will find it within the first 5 s.

6.5 Computer Diplomacy Challenge

Finally, we have also submitted D-Brane to the first Computer Diplomacy Challenge which was part of the 2015 ICGA Computer Olympiad. For this competition we only submitted the non-negotiating version (no implicit agreements and no explicit agreements) of D-Brane, because our negotiation language is not compatible with the DipGame negotiation framework, which was used for the competition.

The competition had three participants: D-Brane, DipBlue, and another non-negotiating player called SuperBot. The agents played a number of games in two different competition set-ups. The first set-up consisted of one instance of each of these participating bots and four instances of a RandomBot (a player that only takes entirely random actions). The second set-up consisted of two instances of each participating bot and one RandomBot.

The results were measured according to two criteria: the number of victories and the number of Supply Centers conquered. The results are displayed in Table 5. We see that D-Brane won the competition by a large difference, both measured in terms of victories and in terms of conquered Supply Centers. This again confirms the previous results that even without negotiations D-Brane is able to outplay negotiating players.

Table 5 Results of the Computer Diplomacy Challenge

	Victories	Conquered Supply Centers
D-Brane	39	933
DipBlue	11	295
SuperBot	2	376
RandomBot	0	143

7 Conclusions and future work

Our experiments make clear that D-Brane plays better than DumbBot, DipBlue and Fabregues' agent. We also see however that our negotiation algorithm only has a relatively small positive effect, and only when applied in combination with implicit agreements. Of course, one could argue that the NB^3 negotiation algorithm itself is not good enough, but we know from experiments presented in [4] that in at least one other domain NB^3 does produce good results.

Apparently, only negotiating joint battle plans for the current round is not enough to really benefit strongly from negotiations. Indeed, we know from personal experience playing Diplomacy that a good player not only negotiates battle plans for the current round, but also looks farther ahead and negotiates future actions. This again confirms our claim that the field of Automated Negotiations should give more attention to the modeling of complex domains, rather than only the development of bargaining strategies.

In [8, 13] negotiations had a stronger positive impact on the respective agents. We think this is because both their agents were implemented by extending the DumbBot with negotiation capabilities, while the DumbBot is not a very strong player. It is likely that negotiations have a much bigger impact on bad players, because good players have less room for improvement.

Furthermore, it is currently still unknown to us why our negotiation algorithm only works well in combination with implicit agreements. We will need to investigate this further in the future.

The most striking result of our experiments, however, is that even when D-Brane does not negotiate it still achieves much better results then DipBlue and Fabregues' agent, which do negotiate. From this we draw the very important conclusion that in some cases *successful negotiation may depend more on the way the underlying domain is tackled, rather than on the applied bargaining strategy*. We therefore argue that future research in the field of Automated Negotiations should put more emphasis on domains where calculating the utility values of potential deals is a complex task. In realistic negotiation settings one simply cannot assume that an explicit representation of the utility functions is given and easy to calculate. Instead, negotiation algorithms should apply more sophisticated forms of reasoning to determine which deals are profitable.

We think that D-Brane will be very important for future negotiations research, because it allows researchers to compare their algorithms not only with the DumbBot but also with our much stronger agent. We have included the strategic component of D-Brane in the BANDANA framework, so that other researchers can implement their own negotiation algorithms on top of the strategic component. This will have two main advantages. Firstly, it will make it easier for new researchers to develop negotiating Diplomacy players as they will not have to waste time on the development of a strategic player. Secondly, if they do have access to another strategic component, it will allow them to compare to what extent their negotiation algorithms depend on the underlying strategic component.

We also think that D-Brane can be very useful for researchers who would like to use Diplomacy to study the topics of Trust, Coalition Formation, and Argumentation Based Negotiation. They could use the strategic module of D-Brane as the basis for their algorithms so that they do not have to build an entirely new agent from scratch.

So far, we have not compared D-Brane in any way with human players. It would be very interesting to see to what extent D-Brane makes moves that are similar to those of expert players and how well it plays against humans. For this however, we need to set up an environment that allows humans to play against software agents and negotiate with them. Also, we would need some tools that allow experts to analyze the moves made by D-Brane afterwards. Part of this infrastructure already exists, but it is currently not ready to perform such experiments. Therefore, we have to defer such experiments to the future.

Furthermore, we are planning to further develop D-Brane's negotiation algorithm so that it may make more sophisticated deals rather than just battle plans for the current round. We will make D-Brane think more steps ahead instead, which will not only improve the strategic play, but will also allow for more interesting deals, because deals may then involve orders made in several rounds, rather then just the current round. Also, we plan to endow the DumbBot with our negotiation algorithm, to see if negotiations in that case have a greater impact.

Finally, we would like to see if our approach for solving COGs indeed can be generalized. We may for example define games in Game Description Language (GDL) [14] so that the strategic component can decompose the game into micro-games at runtime.

Appendix A

Our experiments are largely implemented using the DipGame framework. However, we have implemented a couple of new tools on top of DipGame to make experimentation easier. We have combined these tools into a new extension of DipGame, that we call BANDANA (BAsic eNvironment for Diplomacy playing Automated Negotiating Agents). It includes the following components:

– A new negotiation server.

- A new negotiation language, which we find simpler to use than DipGame's default language.
- A Notary agent, for the implementation of the Unstructured Negotiation Protocol, as explained in Section 4.
- Several example agents, including D-Brane and DumbBot.
- The strategic component of D-Brane.
- Example code that shows how one can implement a negotiating agent on top of D-Brane's strategic component.
- An Adjudicator which, given a set of orders for all units, determines which of those orders are successful.
- A Game Builder, which allows users to set up a customized board configuration. This can be useful for testing.

The BANDANA framework can be downloaded from: http://www.iiia.csic.es/~davedejonge/bandana.

More details about BANDANA's negotiation language and its other tools can be found in the manual which can also be downloaded from the same address.

Appendix B

We here give a more thorough definition of the one-shot game Dip_ϵ. We claim that Dip_ϵ can be modeled as a COG, and that a single round of Diplomacy can be seen as an instance of the negotiation game $N(Dip_\epsilon)$.

Definition 9 Let ϵ denote a configuration of units on the Diplomacy map. Then Dip_ϵ is defined by the tuple:

$$(Gr, SC, Pl^{Dip}, (Units_1, \dots Units_7), loc, (f_1^{Dip_\epsilon} \dots f_7^{Dip_\epsilon})).$$

Here, Gr is a symmetric graph, of which the vertices are called **provinces**. The set of provinces is denoted $Prov$ and we use the notation $adj(p, q)$ to state that provinces p and q are adjacent in the graph. The set of **Supply Centers** SC is a subset of $Prov$. The set Pl^{Dip} represents the 7 players: $Pl^{Dip} = \{\alpha_1, \dots \alpha_7\}$. For each player α_i there is a finite set $Units_i$, which we call the set of **units** owned by α_i. These sets are all disjoint: $i \neq j \Rightarrow Units_i \cap Units_j = \emptyset$. The set of all units is denoted: $Units = \bigcup_{i=1}^{7} Units_i$. The state ϵ of the game implicitly defines an injective function $loc : Units \rightarrow Prov$ that assigns a province (the **location** of u) to any unit u. In order to define the utility functions $f_i^{Dip_\epsilon}$ we first need to define several other concepts.

Given the state ϵ we can define the set of possible **orders** Ord^ϵ:

$$\begin{aligned} Ord^\epsilon &= Mto^\epsilon \bigcup Sup^\epsilon \\ Mto^\epsilon &= \{(u, p) \in Units \times Prov \mid p = loc(u) \vee adj(p, loc(u))\} \\ Sup^\epsilon &= \{(u, u') \in Units \times Units \mid u \neq u'\} \end{aligned}$$

The orders in Mto^ϵ are called **move-to orders**[11] and the orders in Sup^ϵ are called **support orders**. We use the notation Ord_u^ϵ to denote the subset of Ord^ϵ consisting of all possible orders for a given unit u.

$$\begin{aligned} Ord_u^\epsilon &= Mto_u^\epsilon \bigcup Sup_u^\epsilon \\ Mto_u^\epsilon &= \{(u', p) \in Mto^\epsilon \mid u' = u\} \\ Sup_u^\epsilon &= \{(u', u'') \in Sup^\epsilon \mid u' = u\} \end{aligned}$$

Furthermore, we will use Ord_i^ϵ to denote the set possible orders for any unit of player α_i:

$$\begin{aligned} Ord_i^\epsilon &= \bigcup_{u \in Units_i} Ord_u^\epsilon \\ Ord_{-i}^\epsilon &= Ord^\epsilon \setminus Ord_i^\epsilon \end{aligned}$$

and for a set of players $C \subset Pl$ we define:

$$\begin{aligned} Ord_C^\epsilon &= \bigcup_{\alpha_i \in C} Ord_i^\epsilon \\ Ord_{-C}^\epsilon &= Ord^\epsilon \setminus Ord_C^\epsilon \end{aligned}$$

We use similar notation conventions for other sets, such as $Units$, Mto^ϵ and Sup^ϵ.

An action μ_i for a player α_i in Dip_ϵ is then defined as a set of orders, containing exactly one order for each of α_i's units:

$$\mathcal{M}_i^{Dip_\epsilon} = \{\mu_i \subset Ord_i^\epsilon \mid \forall u \in Units_i : |Ord_u^\epsilon \cap \mu_i| = 1\} \tag{4}$$

Definition 10 If α_i plays an action μ_i that contains order o then we say that α_i **submits the order** o. If $\mu = (\mu_1, \mu_2, ...\mu_7)$ is an action profile then $\hat{\mu}$ denotes the set of all orders submitted by all players:

$$\hat{\mu} = \bigcup_{i=1}^{7} \mu_i$$

Definition 11 A support order $(u, u') \in Sup^\epsilon$ is considered **valid**, for an action profile μ, if $\hat{\mu}$ contains a move-to order $(u', p) \in Mto^\epsilon$ where p is adjacent to the location of u. This is denoted by the predicate $val(\mu, u, u')$.

$$val(\mu, u, u') \quad \Leftrightarrow \quad \exists p \in Prov : (u', p) \in \hat{\mu} \ \wedge \ adj(p, loc(u))$$

The rules of Diplomacy specify that players may only submit support orders that are valid.

[11] In this model we consider hold orders as a special kind of move-to order, for which the destination is the current location of the unit: $p = loc(u)$.

Definition 12 Given an action profile μ, a support $(u, u') \in Sup_u^\epsilon$ in $\hat{\mu}$ is said to be **cut** if $\hat{\mu}$ also contains a move-to order that moves to the location of u:

$$cut(\mu, u) \;\Leftrightarrow\; \exists(u, u') \in \hat{\mu} \;\wedge\; \exists(u'', p) \in \hat{\mu} \;\wedge\; p = loc(u)$$

Definition 13 The set of **successful supports** of u in an action profile μ is defined as those orders that support u, and that are valid and not cut:

$$SucSup_{\mu,u} = \{(u', u) \in \hat{\mu} \mid val(\mu, u', u) \wedge \neg cut(\mu, u')\}$$

Definition 14 The **force** $s(\mu, u, p)$ exerted by unit u on province p is defined as:

$$s(\mu, u, p) = \begin{cases} 1.5 + |SucSup_{\mu,u}| & \text{if } (u, p) \in \hat{\mu} \;\wedge\; p = loc(u) \\ 1 + |SucSup_{\mu,u}| & \text{if } (u, p) \in \hat{\mu} \;\wedge\; p \neq loc(u) \\ 0 & \text{otherwise} \end{cases}$$

Definition 15 We say a player α_i **conquers** a province p if α_i has a unit that exerts more force on p than any other unit:

$$conq(\mu, i, p) \Leftrightarrow \exists u \in Units_i \; \forall u' \in Units \setminus \{u\} : s(\mu, u, p) > s(\mu, u', p)$$

Finally, we can define the utility function $f_i^{Dip_\epsilon}$ for a player α_i as the number of Supply Centers he or she conquers:

$$f_i^{Dip_\epsilon}(\mu) = |\{p \in SC \mid conq(\mu, i, p)\}| \tag{5}$$

A natural way to play Diplomacy is to determine for each Supply Center separately whether, and how, it can be conquered. However, the decision how to attack one Supply Center may restrict the possibilities to attack another Supply Center if the same units are involved. This is the essence of a COG. Therefore, we will now show how Dip_ϵ can be modeled as a COG.

We define the set of units of player α_i involved in province p as those units that may move to p, hold in p, support a unit holding in or moving to p, or that may cut any opponent unit that could give support to another opponent unit holding in p or moving to p. This set is denoted by $Units_{p,i}$. More precisely, it consists of all units next to or inside p, and all units located next to an opponent's unit that is located next to p:

Definition 16 The set of units of player α_i **involved in province** p, denoted $Units_{p,i}$ is defined as:

$$\begin{aligned} Units_{p,i} &= MayAttackOrSupport_{p,i} \cup MayCut_{p,i} \\ MayAttackOrSupport_{p,i} &= \{u \in Units_i \mid loc(u) = p \vee adj(p, loc(u))\} \\ MayCut_{p,i} &= \{u \in Units_i \mid \exists u' \in Units_{-i} : adj(loc(u), loc(u')) \quad \wedge \\ &\qquad (loc(u') = p \vee adj(p, loc(u')))\} \end{aligned}$$

We model Dip_ϵ as a COG by defining a micro-game $Dip_{\epsilon,p}$ for each Supply Center $p \in SC$. An action in such a micro-game consists of a set of orders containing maximally 1 order for each unit of player α_i involved in Supply Center p:

$$\mathcal{M}_i^{Dip_{\epsilon,p}} = \left\{ \mu_{p,i} \subset \bigcup_{u \in Units_{p,i}} Ord_u^\epsilon \mid \forall u \in Units_{p,i} : |\mu_{p,i} \cap Ord_u^\epsilon| \leq 1 \right\}$$

Note that in this definition a player is not required to submit an order to each unit involved in p. This is because that unit may instead be used to attack or defend another province p'.

This definition implies that there are binary constraints between the micro-games, because a unit may be involved in more than one province. Two actions $\mu_{p,i}$ and $\mu_{q,i}$ are incompatible if for any unit involved in both provinces, two different orders are submitted. That is, $\mu_{p,i}$ and $\mu_{q,i}$ are compatible iff the following restriction holds:

$$\forall u \in Units_i : |(\mu_{p,i} \cup \mu_{q,i}) \cap Ord_u^\epsilon| \leq 1 \tag{6}$$

We define the utility function of the micro-game $Dip_{\epsilon,p}$ to be:

$$f_i^{Dip_{\epsilon,p}}(\mu_p) = \begin{cases} 1 & \text{if } conq(\mu_p, i, p) \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Here, μ_p is an action profile in the micro-game $Dip_{\epsilon,p}$. The question whether $conq(\mu, i, p)$ holds only depends on the orders submitted for the units involved in p. Therefore, if $\hat{\mu}_p$ is a subset of $\hat{\mu}$ then $conq(\mu_p, i, p)$ holds iff $conq(\mu, i, p)$ holds, which means that (5) and (7) are consistent with (1).

In the following, we use the notation $Units_C$, for a set of players C (a 'coalition') to denote the union of the units of those players:

$$Units_C = \bigcup_{\alpha_i \in C} Units_i$$
$$Units_{-C} = Units \setminus Units_C$$

Definition 17 Let C denote a coalition containing player α_i, then a **battle plan** β for α_i and Supply Center p is a set of orders $\beta \subset Ord^\epsilon$ of the form:

$$\beta = \{(u, p)\} \cup Supports \cup Cuts$$

with $u \in Units_i$, and $Supports \subseteq Sup_C^\epsilon$ is a (possibly empty) set of support orders (u', u) that support u and $Cuts \subseteq Mto_C^\epsilon$ a (possibly empty) set of orders that aim to cut any opponent unit that may support a hostile move into p:

$$(u'', p') \in Cuts \;\; \Rightarrow \;\; \exists u' \in Units_{-C} : \; loc(u') = p' \; \wedge \; adj(p', p)$$

Furthermore we have the restriction that β can contain at most one order for each unit:

$$\forall u \in Units : |\beta \cap Ord_u| \leq 1$$

The set of all battle plans for player α_i on Supply Center p is denoted $B_{i,p}^\epsilon$.

Definition 18 Given a set of deals X, an **invincible plan** is a battle plan $\beta \in B_{i,p}^\epsilon$ for some province p and some player α_i that for every action profile that satisfies X and in which all orders of β are submitted α_i will conquer p. That is, β is invincible iff the following holds:

$$\forall \mu \in \mathcal{M}^{Dip_\epsilon[X]} : \quad \beta \subset \hat{\mu} \quad \Rightarrow \quad conq(\mu, i, p)$$

Furthermore, we can determine all invincible pairs:

Definition 19 Given a set of deals X, an **invincible pair** is a pair of battle plans $(\beta_1, \beta_2) \in B_{i,p}^\epsilon \times B_{i,q}^\epsilon$ for two provinces p, q and some player α_i that guarantees player α_i to either conquer p or conquer q:

$$\forall \mu \in \mathcal{M}^{Dip_\epsilon[X]} : \; \beta_1 \cup \beta_2 \subset \hat{\mu} \; \Rightarrow \; conq(\mu, i, p) \vee conq(\mu, i, q)$$

References

1. An B, Sim KM, Tang L, Li S, Cheng D (2006) Continuous time negotiation mechanism for software agents. IEEE Trans Syst Man Cybern B: Cybern 36(6):1261–1272
2. An B, Gatti N, Lesser V (2009) Extending alternating-offers bargaining in one-to-many and many-to-many settings. In: Proceedings of the 2009 IEEE/WIC/ACM international joint conference on web intelligence and intelligent agent technology—volume 02. IEEE Computer Society, Washington, WI-IAT '09, pp 423–426. doi:10.1109/WI-IAT.2009.188
3. Baarslag T, Hindriks K, Jonker CM, Kraus S, Lin R (2010) The first automated negotiating agents competition (ANAC 2010). In: Ito T, Zhang M, Robu V, Fatima S, Matsuo T (eds) New trends in agent-based complex automated negotiations, series of studies in computational intelligence. Springer, Berlin
4. de Jonge D, Sierra C (2015) NB3: a multilateral negotiation algorithm for large, non-linear agreement spaces with limited time. Auton Agent Multi-Agent Syst 29(5):896–942. doi:10.1007/s10458-014-9271-3. http://www.iiia.csic.es/files/pdfs/jaamas%20NB3.pdf
5. Dechter R, Mateescu R (2007) And/or search spaces for graphical models. Artif Intell 171(2–3):73–106. doi:10.1016/j.artint.2006.11.003. http://www.sciencedirect.com/science/article/pii/S000437020600138X
6. Dimopoulos Y, Moraitis P (2011) Advances in argumentation based negotiation. In: Negotiation and argumentation in multi-agent systems: fundamentals, theories systems and applications, pp 82–125
7. Endriss U (2006) Monotonic concession protocols for multilateral negotiation. In: Proceedings of the fifth international joint conference on autonomous agents and multiagent systems. ACM, New York, AAMAS '06, pp 392–399. doi: 10.1145/1160633.1160702
8. Fabregues A (2014) Facing the challenge of automated negotiations with humans. PhD thesis, Universitat Autònoma de Barcelona
9. Fabregues A, Sierra C (2011) Dipgame: a challenging negotiation testbed. Engineering Applications of Artificial Intelligence
10. Faratin P, Sierra C, Jennings NR (1998) Negotiation decision functions for autonomous agents. Robot Auton Syst 24(3-4):159–182. doi:10.1016/S0921-8890(98)00029-3. http://www.sciencedirect.com/science/article/pii/S0921889098000293, multi-Agent Rationality
11. Faratin P, Sierra C, Jennings NR (2000) Using similarity criteria to make negotiation trade-offs. In: International conference on multi-agent systems, ICMAS'00, pp 119–126
12. Fatima S, Wooldridge M, Jennings NR (2009) An analysis of feasible solutions for multi-issue negotiation involving nonlinear utility functions. In: Proceedings of the 8th international conference on autonomous agents and multiagent systems—volume 2, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, AAMAS '09, pp 1041–1048, http://dl.acm.org/citation.cfm?id=1558109.1558158
13. Ferreira A, Lopes Cardoso H, Paulo Reis L (2015) Dipblue: a diplomacy agent with strategic and trust reasoning. In: 7th international conference on agents and artificial intelligence (ICAART 2015), pp 398–405
14. Genesereth M, Love N, Pell B (2005) General game playing: overview of the aaai competition. AI Mag 26(2):62–72
15. Hemaissia M, El Fallah Seghrouchni A, Labreuche C, Mattioli J (2007) A multilateral multi-issue negotiation protocol. In: Proceedings of the 6th international joint conference on autonomous agents and multiagent systems, ACM, New York, AAMAS '07, pp 155:1–155:8. doi: 10.1145/1329125.1329314
16. Ito T, Klein M, Hattori H (2008) A multi-issue negotiation protocol among agents with nonlinear utility functions. Multiagent Grid Syst 4:67–83. http://dl.acm.org/citation.cfm?id=1378675.1378678
17. Klein M, Faratin P, Sayama H, Bar-Yam Y (2003) Protocols for negotiating complex contracts. IEEE Intell Syst 18(6):32–38. doi:10.1109/MIS.2003.1249167

18. Koenig S, Tovey C, Lagoudakis M, Markakis V, Kempe D, Keskinocak P, Kleywegt A, Meyerson A, Jain S (2006) The power of sequential single-item auctions for agent coordination. In: Proceedings of the AAAI conference on artificial intelligence (AAAI), pp 1625–1629
19. Kraus S, Lehmann D (1995) Designing and building a negotiating automated agent. Comput Intell 11:132–171
20. Kraus S, Lehman D, Ephrati E (1989) An automated diplomacy player. In: Levy D, Beal D (eds) Heuristic programming in artificial intelligence: the 1st computer Olympia. Ellis Horwood Limited, pp 134–153
21. Krishna V, Serrano R (1996) Multilateral bargaining. Rev Econ Stud 63(1):61–80. http://EconPapers.repec.org/RePEc:bla:restud:v:63:y:1996:i:1:p:61-80
22. Lai G, Sycara K, Li C (2008) A decentralized model for automated multi-attribute negotiations with incomplete information and general utility functions. Multiagent Grid Syst 4:45–65. http://dl.acm.org/citation.cfm?id=1378675.1378677
23. Marsa-Maestre I, Lopez-Carmona MA, Velasco JR, de la Hoz E (2009) Effective bidding and deal identification for negotiations in highly nonlinear scenarios. In: Proceedings of the 8th international conference on autonomous agents and multiagent systems—volume 2. International Foundation for Autonomous Agents and Multiagent Systems, Richland, AAMAS '09, pp 1057–1064. http://dl.acm.org/citation.cfm?id=1558109.1558160
24. Marsa-Maestre I, Lopez-Carmona MA, Velasco JR, Ito T, Klein M, Fujita K (2009) Balancing utility and deal probability for auction-based negotiations in highly nonlinear utility spaces. In: Proceedings of the 21st international joint conference on artifical intelligence. Morgan Kaufmann Publishers Inc., San Francisco, IJCAI'09, pp 214–219. http://dl.acm.org/citation.cfm?id=1661445.1661480
25. Nash J (1950) The bargaining problem. Econometrica 18:155–162
26. Nash J (1951) Non-cooperative games. Ann Math 54(2):286–295. http://www.jstor.org/stable/1969529
27. Nguyen TD, Jennings NR, 2004 Coordinating multiple concurrent negotiations. In: Proceedings of the third international joint conference on autonomous agents and multiagent systems - volume 3. IEEE Computer Society, Washington, AAMAS '04, pp 1064–1071. doi:10.1109/AAMAS.2004.94
28. Poundstone W (1993) Prisoner's Dilemma, 1st edn. Doubleday, New York
29. Rosenschein JS, Zlotkin G (1994) Rules of encounter. MIT Press, Cambridge
30. Rossi F, Beek PV, Walsh T (2006) Handbook of constraint programming (foundations of artificial intelligence). Elsevier Science Inc., New York
31. Serrano R (2008) Bargaining. In: Durlauf SN, Blume LE (eds) The new Palgrave dictionary of economics. Palgrave Macmillan, Basingstoke
32. Sierra C, Jennings NR, Noriega P, Parsons S (1997) A framework for argumentation-based negotiation. In: International workshop on agent theories, architectures, and languages. Springer, pp 177–192
33. Sycara KP (1990) Persuasive argumentation in negotiation. Theor Decis 28(3):203–242. doi:10.1007/BF00162699

Dave de Jonge completed his PhD in Computer Science at the Autonomous University of Barcelona in 2015. He received a Master's degree in Mathematical Physics in 2008 from the University of Amsterdam. His interests are in Autmated Negotiations, Game Theory, Constraint Optimization, Logic, and General Game Playing. His work currently focusses on the application of Automated Negotiations to complex extensive form games.

Carles Sierra is Vice-Director of the Artificial Intelligence Research Institute (IIIA) of the Spanish National Research Council (CSIC) located in the area of Barcelona. He has been contributing to Artificail Intelligence research since 1985 in the areas of Knowledge Representation, Auctions, Electronic Institutions, Autonomous Agents and Multiagent Systems, and Agreement Technologies. He has participated in more than 20 EU funded projects, is or has been member of several journal editorial boards including AIJ and JAIR and is the editor-in-chief of the JAAMAS journal. He organised IJCAI in 2011 in Barcelona and is the Program Chair of IJCAI 2017. He is an EurAI Fellow.