Strategic Negotiation and Trust in Diplomacy – The DipBlue Approach

André Ferreira[1], Henrique Lopes Cardoso[1,2], and Luís Paulo Reis[2,3]

[1] Dep. Eng. Informática, Faculdade de Engenharia, Universidade do Porto, Porto, Portugal
[2] LIACC – Laboratório de Inteligência Artificial e Ciência de Computadores, Porto, Portugal
[3] DSI/EEUM – Escola de Engenharia da Universidade do Minho, Portugal
andre.ferreira.v2@gmail.com, hlc@fe.up.pt, lpreis@dsi.uminho.pt

Abstract. The study of games in Artificial Intelligence has a long tradition. Game playing has been a fertile environment for the development of novel approaches to build intelligent programs. Multi-agent systems (MAS), in particular, are a very useful paradigm in this regard, not only because multi-player games can be addressed using this technology, but most importantly because social aspects of agenthood that have been studied for years by MAS researchers can be applied in the attractive and controlled scenarios that games convey. Diplomacy is a multi-player strategic zero-sum board game, including as main research challenges an enormous search tree, the difficulty of determining the real strength of a position, and the accommodation of negotiation among players. Negotiation abilities bring along other social aspects, such as the need to perform trust reasoning in order to win the game. The majority of existing artificial players (bots) for Diplomacy do not exploit the strategic opportunities enabled by negotiation, focusing instead on search and heuristic approaches. This paper describes the development of *DipBlue*, an artificial player that uses negotiation in order to gain advantage over its opponents, through the use of peace treaties, formation of alliances and suggestion of actions to allies. A simple trust assessment approach is used as a means to detect and react to potential betrayals by allied players. DipBlue was built to work with DipGame, a MAS testbed for Diplomacy, and has been tested with other players of the same platform and variations of itself. Experimental results show that the use of negotiation increases the performance of bots involved in alliances, when full trust is assumed. In the presence of betrayals, being able to perform trust reasoning is an effective approach to reduce their impact.

Keywords: Diplomacy, Strategy, Negotiation, Trust

1 Introduction

Since the beginning of Artificial Intelligence as a research field, game playing has been a fertile environment for the development of novel approaches to build intelligent machines. Besides puzzles and 2-player games (such as chess or checkers), for which contributions based on search and heuristics have been much successful, multi-player games make it harder to develop winning strategies. Multi-agent systems (MAS) are a useful paradigm for modeling such games, because of their natural fit into these scenarios. Furthermore, applying MAS research in games opens the possibility of applying social aspects of agenthood, which have been studied for years by MAS researchers, in the attractive and controlled scenarios that games convey.

Most approaches to game playing have been based mainly on (adversarial) search techniques and sophisticated domain-specific heuristics. Complex adversarial multi-player games pose new challenges to MAS research, given the fact that their search spaces are big enough to render ineffective any (current) approach based solely on search and heuristics. Of particular interest are those games in which a social dimension is included [14].

An example of the latter is Diplomacy, a military strategy multi-player simultaneous move board game, created by Allan B. Calhamer [1] in 1954. Its most interesting attributes include, according to Hall and Loeb [6], the enormous size of its search tree, the difficulty of determining the real strength of a position, and negotiation, whose support leverages the development of sophisticated players with an important competitive advantage. The fact that adversaries may negotiate throughout the game makes Diplomacy a very appealing sandbox for multi-agent research: while players are competing against each other, they must also cooperate to win the game. To do so, players may need to build trust, maintain relationships and negotiate deals, for which they may use argumentation techniques.

This work proposes an approach to the creation of a Diplomacy artificial player that takes advantage of negotiation and trust in order to enhance its performance. Our efforts focus on showing its competitive advantage, that is, on showing that by making use of social skills we are able to obtain an agent that is capable of surpassing its opponents. Our bot, *DipBlue*, works with the MAS testbed *DipGame* [4] and has been tested with another player of the same platform and with variations of itself (DipBlue archetypes).

The rest of the paper is structured as follows. Section 2 briefly describes the rules of Diplomacy and highlights the properties of the game that make it appealing for MAS research. Section 3 reviews related work on Diplomacy platforms and bots. In Section 4 we describe DipBlue's architecture and archetypes. Section 5 puts forward an experimental evaluation of DipBlue, presenting and discussing the obtained results. In Section 6 we draw some conclusions of the work done, and we point out some directions for future work.

2 Diplomacy: The Game

Diplomacy falls into the category of social games that explicitly allow collusion, a strategy deemed illegal in many real-world situations, such as markets and auctions. By including negotiation and allowing teamwork, Diplomacy offers the possibility of employing social skills when building intelligent playing agents, such as strategic negotiation, opponent modeling and trust reasoning.

Diplomacy action takes place in the beginning of the 20th century, in the years before World War I. Each player represents one of the following countries or world powers: England, France, Austria, Germany, Italy, Turkey and Russia. The main goal of the game is to conquer Europe, which is achieved by acquiring a minimum of 18 from a total of 34 *supply centers* throughout the map (see Figure 1). During the game, each player commands its units in the map by giving them orders to *hold*, *move* to (i.e., *attack*) adjacent regions, or *support* other units actions (holds or moves of units from either the same or other players). Move actions to occupied regions originate conflicts (*standoffs*): the strongest unit (attacker or defender) wins a standoff, where strength is increased by backing up units with supports from other neighboring units. Some moves may invalidate other moves or cut supports. It is the conjunction of all orders that determines what actually happens in each round of the game[4].

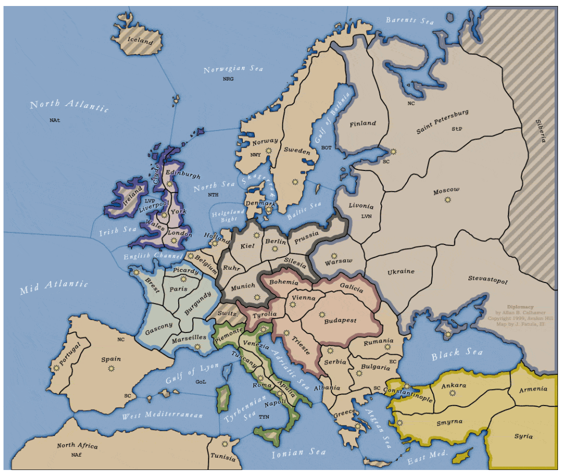

Fig. 1. Standard Diplomacy map of Europe

Before each round of orders, players are able to communicate freely with each other in a negotiation phase. They can do so both publicly and privately, with the aim of establishing transient agreements. Although these conversations and arrangements are a huge part of the game-play (particularly in human tournaments), they hold absolutely no real power in the game itself: a player can commit to execute an action in exchange of information and decide not to fulfill its part of the agreement once its individual goal has been achieved. Collective

[4] Detailed rules of the game can be found in [1].

goals are thus always short-lived, given the most prominent individual goal of conquering the world.

Diplomacy is characterized by having no random factors (besides the initial assignment of world powers to players) and being a zero-sum game (as far as the final outcome of the game is concerned). However, the size of the game's search tree is enormous and impossible to search systematically even at low depths. In most other games, in order to address this problem the tree is pruned using heuristics that assess the state of the game at a given time and compare it with future game states. However, this technique cannot be directly applied to Diplomacy, given the fact that the game is played in a multi-agent partially observable environment [19], and thus not fully-deterministic from an agent's point of view – the chosen orders of a player are not necessarily effective, given its lack of knowledge about other agents' actions.

Solving problems by searching consists of representing the problem at hand as a search space (typically a graph) and employing a search technique to find a path through the graph. Heuristic search approaches (e.g. branch and bound, A*) try to take advantage of domain information to guide this process. Handling adversarial games consists of taking into account that certain moves in the path will be taken by the opponent, therefore building alternate decision layers in the search tree – the player does not have full control of the course of the game. Algorithms such as minimax and its variations are appropriate for handling these alternating moves kind of games, but still require the use of good heuristics to evaluate game states as accurately as possible.

Applying this kind of algorithms to Diplomacy is particularly challenging, not only because of its large state space, but also because it is hard to define appropriate game state evaluation functions. When attempting to create heuristics for Diplomacy (e.g. [6, 20]), a player can be overlooked as a weak opponent when considering only the number and placement of its armies; and yet, when having strong alliances, a player can win the game or annihilate another player in a few turns. This makes the creation of an effective heuristic a difficult challenge.

Effective Diplomacy players need therefore to be developed using alternative (or complementary) approaches. The rich environment provided by Diplomacy promotes the development of bots capable of dominating their opponents through *negotiation*, which increases the need for *trust* reasoning capabilities to allow players to protect themselves.

3 Related Work

Sophisticated agent models have been developed over the years, including cognitive models for negotiation, argumentation, trust and reputation reasoning. In order to compare different advances in multi-agent systems, shared domains and rich environments are needed, providing challenging scenarios in which researchers can test their models. The Diplomacy game has been argued to provide such a testbed [3, 5], adding also the possibility of putting together humans and agents in a complex interacting environment.

In fact, the Diplomacy game has been studied for a long time within the MAS research community. One of the first attempts to create a software agent for this game dates back to more than 25 years ago, by Kraus et al. [12]. Agent theories and architectures have been applied when developing Diplomacy agents. For example, Krzywinski *et al.* [14] have developed their Diplomacy agent following the well-known subsumption architecture.

Given the wide availability of human-based Diplomacy tournaments[5], some authors (such as Kemmerling *et al.* [10]) have devoted attention to develop human-like playing behavior, for which opponent modeling techniques play an important role. Developing agents that learn autonomously to play Diplomacy has also received attention from the research community. Shapiro *et al.* [20] have used temporal-difference learning and self-play to automatically develop a playing strategy, in an attempt to overcome the limitations of search-based approaches in the tremendous search space Diplomacy has. However, they focused in a non-cooperative version of the game, meaning that negotiation does not take place. On the other hand, Kemmerling *et al.* [11] applied an evolutionary algorithm-based planning approach for enhancing the performance of a Diplomacy agent that includes negotiation capabilities. However, given their bias on enhancing a believable bot, performance against the Albert benchmark (see below) is a bit weak. An optimization of a Diplomacy bot through genetic algorithms has also been done by de Jonge [9].

A short description of testbeds for Diplomacy follows, together with popular automated agents (bots) developed specifically for this game. We also review some of the main strategies used in the game, both in terms of evaluation heuristics and negotiation.

3.1 Diplomacy Testbeds

Although there are several different testbeds for MAS in general, there are a few specific for Diplomacy. The two most influential are briefly described here.

The Diplomacy Artificial Intelligence Development Environment (DAIDE[6]) assists the development of Diplomacy bots by taking care of all the logic concerning moves validation and the generation of resulting game states. It also provides a communication server that allows players to exchange messages between them during certain phases of the game. This communication server provides several layers of supported syntax in a way to allow for both simpler and more complex negotiating bots. The communication layers are referred to as *Press levels* and there are 15 distinct ones, ranging from the most basic (no communication at all) to the more complex level, in which free text negotiation is enabled. Both server and bots are written in C/C++.

DipGame[7] [5] is a testbed created at IIIA-CSIC that uses the DAIDE server to handle moves resolution and generation of new game states. Although DAIDE

[5] See e.g. `http://www.playdiplomacy.com/`

[6] `http://www.daide.org.uk/`

[7] `http://www.dipgame.org/`

already supports communication, DipGame introduces its own server and creates a new communication syntax known as L Language. This language is composed of 8 levels, ranging from level 1 concerning deal negotiation, to level 7 foreseeing the use of argumentation. DipGame and its bots are implemented in Java. Additionally, DipGame provides an improved logging system and a web interface where anyone can play against some DipGame bots.

3.2 Diplomacy Bots

Some popular and relevant bots developed for Diplomacy are briefly mentioned here. These bots have different approaches, and some of them have been used as an inspiration during the creation of DipBlue.

Israeli Diplomat [12, 13] was arguably the first attempt to create an automated Diplomacy player, by Kraus *et al.*. It uses an architecture that distributes responsibilities according to the nature of the tasks. This architecture has served as an inspiration for other bots, such as the Bordeaux Diplomat. Israeli Diplomat has several well designed strategies to deal with heuristic search and negotiation.

The Bordeaux Diplomat [15] was created by Loeb and has a partitioned structure like the Israeli Diplomat, separating negotiation from solution search. The latter ignores the world power that owns each region and does an impartial evaluation of sets of actions by using a best first algorithm. The bot keeps a social relations matrix to determine the opponents that are more likely to betray.

DumbBot [16] is probably the most popular and common bot available for DAIDE. Even though it is not optimized and performs only a small tactical analysis, DumbBot performs relatively well, beating some attempts to create complicated heuristics and tactics. It does not perform negotiation of any sort – the only actions made are game-related orders. The bot has been the target of many studies and has been used as a benchmark for testing other bots. A replica of DumbBot was developed for DipGame [9], different only on the lack of support for a move called Convoy, which is not available in DipGame.

The Albert [21] bot was developed by van Hal and is, up until now, the best bot for DAIDE by far. It is the only Press Level 30 bot available. Because of its efficiency and high performance, it has been used as a benchmark by many researchers who try to outperform it.

BlabBot was created by Newbury [22] and builds on DumbBot by adding negotiation capabilities to it. BlabBot follows a "peace-to-all" strategy by sending peace offers to all players, decreasing the value of regions owned by players accepting those peace offers.

DarkBlade [18] is a no-press bot built by Ribeiro, which tries to combine the best tactics and strategies used by other Diplomacy agents. DarkBlade follows a modular architecture similar to Israeli Diplomat, and is modeled as an internal MAS, using so-called sub-agents.

HaAI [8] was developed by Johansson and Håård, and uses a MAS structure inside the bot itself, in which each unit owned by the player is represented as an individual sub-agent. Each sub-agent tries to choose its own action according to what it considers to be the best option, while at the same time interacting as a team with the other sub-agents of the same player.

SillyNegoBot [17] is a DipGame bot developed by Polberg *et al.* and is an extension to the SillyBot, a bot similar to DumbBot. SillyNegoBot adds L Language level 1 communication and includes a BDI architecture. The bot has proven to be successful when matched with DumbBot but too naive when confronted with betrays. It uses the concept of personality with ratios for aggression/caution.

A few other works worth mentioning include an approach to optimize a Diplomacy bot using genetic algorithms [9], and a bot that takes advantage of a moves database, based on abstract state templates, providing the best set of actions for a given map and units with the goal of acquiring certain regions [2].

3.3 Strategies for Diplomacy

Evaluating board positions is crucial for effective Diplomacy playing. However, board evaluation is particularly complex in Diplomacy, both because of the partially observable environment a player is facing and the potential use of negotiation to establish temporary alliances between players. Had a player access to the private deals its opponents establish, a much more precise evaluation would be possible. Good references that describe strategies for Diplomacy include [6] and [22], besides all the information available online in sites such as DAIDE's. We limit ourselves to describe a few of the most often used ones.

The *province destination value* is used by DumbBot to assign a value to each region [9]. This metric takes into account the player that owns the region, and the amount of allied and enemy units in surrounding regions. The *blurred destination value* is a variation of the previous metric that spreads the value of a certain node to its neighbors. This way, the surrounding regions reflect that either the region itself is valuable or is near a valuable region. Values assigned to near regions can be obtained in a number of ways, e.g. by applying a Gaussian or linear blur.

Negotiation strategies often used in Diplomacy try to limit the search space by establishing cooperation agreements among players. However, when time comes such agreements may be simply ignored, and betrays come into play. This is why the establishment of an alliance does not *per se* comprise a real enhanced power to the players: the competitive advantage obtained by negotiating an agreement is based on the assumption of compliance, and thus any agreement is unstable in a zero-sum game like Diplomacy. Some of the main negotiation tactics that have been proposed in Diplomacy literature are briefly mentioned here. Many of these tactics are used by human players in real board games. However, they typically use concepts that are simple for humans but complicated for computers, such as small clues gathered just by looking at the opponents and the confidence the player has on other players.

The *peace-to-all* strategy is used in BlabBot, and tries to provide a certain level of security by quickly establishing alliances [22]. Players outside this set of alliances have a high chance of being eliminated, and the bot will progressively betray the player that is considered the most convenient to leave the allied group – usually the strongest one.

Back-stab is a tactic used by BlabBot for deciding when to betray alliances or for guessing when these will be betrayed by adversaries [22]. This tactic consists of keeping a threat matrix between the player and the opponents (and vice-versa): the higher the value, the more likely the player is to betray an alliance.

The *power-cluster* strategy is used to determine what world powers the player should ask for alliances and which ones to keep the longest. This strategy has evolved using clustering techniques over several games in order to identify which groups of powers have higher probability of succeeding, when allied.

4 DipBlue

In this section we present our own Diplomacy bot. DipBlue[8] is an artificial player built with the purpose of assessing and exploring the impact of negotiation in a game that natively relies on communication. Since the main difficulty when creating a Diplomacy bot is the size of the search tree, we have chosen to favor negotiation as DipBlue's main tool to gain advantage over its competitors, complemented with a simple form of trust reasoning to detect and react to betrayals.

4.1 Architecture

When designing DipBlue, we aimed at a flexible and easily extendible architecture. For that, a highly modular approach has been used, in which each module evaluates and determines, from its own perspective, the set of orders to apply in each turn. Figure 2 shows a class diagram comprising an overview of DipBlue's architecture, including two main components: Negotiator and Adviser (the latter is further explained in Section 4.3). Different advisers may be added as needed to the bot, exploiting its extensibility. This modular implementation also allows an easy customization of the bot, resulting in a vast array of possible configurations of bots that differ in their capabilities and behaviors. In Section 4.4 we discuss some of such configurations.

Figure 2 also shows the relation between one of the advisers and DumbBot, the bot it is based on. In other words, DumbBot could, in principle, be thought of DipBlue configured with a single adviser: MapTactician (see also Section 4.3).

The negotiation capability of DipBlue is materialized in the Negotiator component, responsible for handling received messages and for determining which messages are to be sent. Negotiation tactics are included in this component. The actual orders to be executed by each of the player's units, however, are dictated by Advisers. Any negotiated agreements that are to have an effect in further DipBlue actions need thus to be taken into account by some advisers (e.g. AgreementExecutor and WordKeeper in Figure 2).

[8] DipBlue is named in honor of the chess-player supercomputer DeepBlue, and of the platform it is built to play on, DipGame.

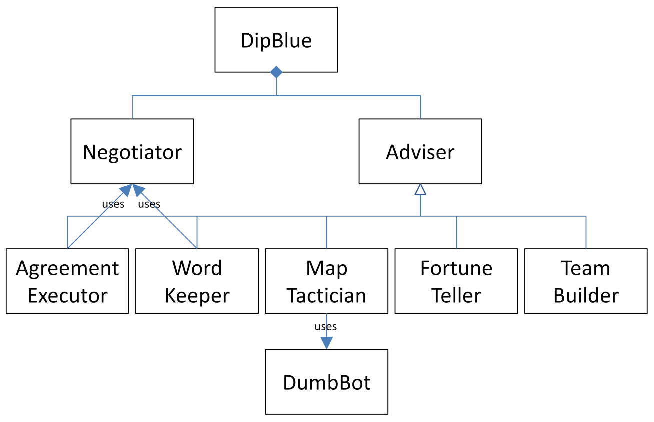

Fig. 2. DipBlue architecture

4.2 Negotiation and Trust

DipBlue is a negotiating bot with the ability to communicate in L Language level 1, whose format is explained in [5]. This layer of the language allows for three types of requests: *peace*, *alliance* and *order requests*.

Peace requests. The most basic strategy of DipBlue consists of reducing friction with its opponents, in an attempt to make the game more predictable. For that, it makes use of peace requests, which reflect the intention for truce to occur among players and can be understood as a request for cease-fire or simply to achieve neutrality. In order to reduce the probability of conflict with as many players as possible, peace messages are sent to all negotiating players in the beginning of the game. DipBlue then opts to break truce with the player considered to be the least beneficial, taking into account the number of supply centers held by the other powers and the proximity between the power under analysis and DipBlue in the map.

Alliance requests. In a more tactical level, alliance requests are handled by using two clusters of powers – allies and enemies – with the purpose of joining the efforts of the allied powers in order to defeat the enemies. DipBlue sends alliance requests to all players with whom it is in a state of peace, targeting the strongest non-ally power as an enemy. This results in a joint effort to eliminate the biggest threat at each phase of the game. Once the previously targeted enemy is weakened enough, the new strongest non-ally power is targeted, and so on. DipBlue accepts alliance requests from other players if they are in a state of peace and if the targeted enemy is not an ally itself.

Order requests. Finally, at the operational level, an order request contains an order regarding a unit of the player to whom the request is sent. It has the purpose of suggesting orders for the receiving player's units. DipBlue uses these messages as a way to request for additional support to moves adjacent to allied units. Since the L Language supports messages with negative connotation, players can ask their allies not to perform actions that interfere with their own. DipBlue accepts order requests if the sender is an ally and if the requested order has a value higher than the action DipBlue had selected for the envisaged unit (see Section 4.3).

Trust reasoning. Orthogonal to the use of the negotiation strategy described above is the maintenance of a *trust ratio* reflecting the relationship between the player and each opponent. Initially all players are neutral, meaning they have a trust ratio of 1. This ratio is converted into a friction ratio $Friction = 1/Trust$, used by the bot to decide on making alliances or to adjust the likelihood of fulfilling deals. It also determines when certain deals are accepted or rejected. The value of orders requested by other players is scaled with the trust ratio of the sender – players with a higher trust ratio have a higher probability of having their requests accepted.

Trust (or friction) ratios are updated during the course of the game. Events that decrease trust (and thus increase friction) include attacks and betrayals. Likewise, the lack of attacks by players in close distance or the fulfillment of agreements bring an increase on trust (and thus a decrease on friction). The magnitude of the impact of these events on trust depends on the current trust held by the player: trust in currently untrustworthy players is less affected; on the other hand, trustworthy players get a higher impact on their assigned trust value. This choice is meant to capture the competitive nature of the game of Diplomacy, emphasizing the role of betrayals during the game. An attack made by an ally (a currently trustworthy opponent) has a higher increase of friction than the same attack made by a current enemy. Given the nature of alliances in Diplomacy, which are not on solid ground and may suddenly be broken, with this approach we try to quickly capture such changes in the game.

Along with the trust ratio, a *state* that also reflects the current relationship is associated with each opponent. This state is originally *neutral* and may change to *war* or *peace* according to the trust ratio and the outcome of negotiations (namely peace and alliance requests). This state is used to enhance the impact of the trust ratio, by increasing its effect when assessing actions related with a given opponent. When a new alliance is started, all enemy player states are changed to war, thus reducing their trust ratio and increasing aggressiveness towards them.

4.3 Advisers

Advisers are the components of DipBlue that assess possible orders and determine what to do. Each adviser is independent, meaning that it can be used without the others, providing modularity and extensibility to the architecture. In the process of determining which actions to perform, the opinions of all employed advisers are taken into account.

A ranking of possible orders for each unit is created. In order to calculate the value assigned to each possible action, we use a weighted accumulation similar to a voting system, considering the numerical evaluation each adviser provides (see Eq. 1, where n is the number of advisers, w^i is the weight of adviser i and v^i_{Order} is the value Adviser i assigns to $Order$).

$$V_{Order} = \sum_{i=1}^{n} w^i . v^i_{Order} \tag{1}$$

While accumulating values, these can actually be either summed or multiplied, as for some advisers the assigned value has no meaning by itself (e.g. the probability of an order being successful), and should be interpreted as a scaling factor – the adviser is simply increasing or decreasing the importance of the order. This also means that the order of application of each adviser evaluation is important.

Finally, the best order for each unit is selected, ensuring they do not collide with each other. This verification is important because, for instance, if two units happen to attack the same region, a conflict arises and neither unit is successful, nullifying each other moves.

Initially, advisers have equal weights, which can then be adjusted in order to fine-tune the bot. Along with these weights, advisers themselves have intrinsic parameters that can be adjusted for obtaining different behavior variations. The adjustment of these parameters allows the creation of behavioral archetypes and personality, such as aggressive, naive, friendly or vengeful players. An optimization approach may be used to find out the optimal performance, following the approach in [9].

We now provide short descriptions of the advisers illustrated in Figure 2.

MapTactician is the base adviser, serving as a starting point for all the following advisers to work upon. It is based on the behavior of DumbBot (see Section 3.2). This adviser performs an assessment of the map in terms of raw power, amount of enemy units and their positions, following a province destination value heuristic (see Section 3.3).

FortuneTeller takes into account the basic rules for resolving actions in Diplomacy to predict if an action will succeed, giving a probabilistic view of the evaluated move actions. Since Diplomacy has a complex set of rules with many exceptions and precedences between them, determining whether one action in a given set is going to be successful is not a trivial task. Given the branching factor of the search tree, it can also be quite time consuming. In order to alleviate this problem, FortuneTeller disregards the possibility of chain actions that may nullify each other, and thus often obtains optimistic probabilities of success.

The role of *TeamBuilder* is to promote support actions. Supports related with move actions that are highly ranked have their value increased, as a way to increase the probability of success of the move. Further in the process of choosing the actions for each unit, with this adviser a unit may abandon its highest ranked action to support instead some neighbor with a high need for support, particularly when the move of such neighbor has a value higher than the original action of the supporting unit. Increasing the weight of this adviser results in a higher cooperation in attacking moves, thus enhancing team play.

AgreementExecutor takes into account the deals made by DipBlue and decides how they should be performed. The value of each deal is assessed by taking into account the trust ratio of the deal's counterpart. Given the dynamics of the game, a deal may be proposed or accepted when the powers are in a friendly state but then be poorly rated because of a decrease of trust between both parties.

WordKeeper is the adviser in charge of reflecting the influence of trust/friction regarding each opponent. WordKeeper scales the value of the actions according to the trust ratio of the player the action is directed to. This way, the value associated with an attack to an ally is reduced, while the value associated with an attack to an enemy is increased.

4.4 Archetypes

Throughout the development of the DipBlue bot some distinct aspects were created, such as the ability to negotiate, propose deals and perform trust reasoning. In order to test some of these aspects individually, we have configured different DipBlue instances according to generic *archetypes*. Each archetype is defined by the set of advisers it uses and by the way the bot reacts to certain events, such as peace and action requests. Archetypes can be seen as different configurations of DipBlue, and were also defined to overcome the lack of DipGame bots available for testing purposes. With the exception of NoPress (see below), every other archetype described below uses all the advisers presented in Section 4.3.

Advisers are used as shown in Eq. 2. For the experiments reported in Section 5, all adviser weights have been set to 1. TeamBuilder, WordKeeper and FortuneTeller are used as scaling factors for the values obtained by MapTactition and AgreementExecutor.

$$\begin{aligned} V = ((((1 + MapTactician + AgreementExecutor) \\ \times TeamBuilder) \\ \times WordKeeper) \\ \times FortuneTeller) \end{aligned} \tag{2}$$

NoPress is the most basic version of DipBlue. It does not perform negotiation nor trust reasoning of any kind. It is very similar to DumbBot in terms of capabilities. NoPress makes use of the MapTactician adviser.

Slave has the ability to communicate, although it does not take the initiative to start negotiations. Slave makes the same evaluation of actions as NoPress, automatically accepts every requests and follows them blindly (as long as they are executable). All agreements have higher priority as compared to the actions determined by the bot itself. This is the best bot to have as an ally.

Naive is endowed with the ability to propose deals of any supported kind to other players. When receiving incoming requests it has the ability to reason whether it should accept them based on a simple evaluation of both the request and the requesting player. Deals proposed by allies or players with very high trust ratio are likely to be accepted, while requests made by players the bot is in war with are almost always rejected. However, Naive lacks the ability to perceive when agreements are not fulfilled, and thus cannot be said to perform trust reasoning.

DipBlue is the more complete bot: it has the same setting as Naive with the addition of being able to perform trust reasoning. This allows DipBlue to detect hostile actions from other players and to assess how they fulfill agreements. Due to the use of trust ratios and both the AgreementExecutor and WordKeeper advisers, DipBlue is also capable of betraying other players. In Algorithm 1 a high-level specification of DipBlue's operation is listed.

Algorithm 1 DipBlue's high-level algorithm

```
Require: $gameState$ {current state of the game}
    $\mathcal{A}$ {advisers to use}
    $\mathcal{X}$ {list of opponents}
    $\mathcal{P}$ {list of opponents in peace and their friction ratios}
    $\mathcal{W}$ {list of opponents in war and their friction ratios}
 1: for all $op \in \mathcal{X}$ do
 2:   $negotiatePeaceAgreement(op, \mathcal{P})$
 3: end for
 4: while $alive$ do
 5:   switch ($phase(gameState)$)
 6:   case $Spring$, $Fall$:
 7:     $updatePeaceAgreements(\mathcal{P})$
 8:     $hp \leftarrow highestPower(gameState)$
 9:     $negotiateAlliance(hp, \mathcal{P}, \mathcal{W})$
10:     $\mathcal{O} \leftarrow selectMoveOrders(gameState, \mathcal{A})$
11:     $requestSupports(\mathcal{O}, \mathcal{P})$
12:   case $Summer$, $Autumn$:
13:     $\mathcal{O} \leftarrow selectRetreatOrders(gameState, \mathcal{A})$
14:   case $Winter$:
15:     $\mathcal{O} \leftarrow selectBuildOrRemoveOrders(gameState, \mathcal{A})$
16:   end switch
17:   $executeOrders(gameState, \mathcal{O})$
18:   for all $op \in \mathcal{X}$ do
19:     for all $o \in executedOrders(gameState, op)$ do
20:       if $isMoveTo(o)$ and $target(o) = me$ then
21:         $updateRatio(op, \mathcal{P}, \mathcal{W})$
22:       end if
23:     end for
24:   end for
25: end while
```

As mentioned in Section 4.2, the bot starts by proposing peace agreements to all adversaries (lines 1-3), and according to received responses updates the set $\mathcal{P}$ of opponents that are in peace.

When playing Diplomacy, in each season the players go through different phases, in the following sequence: spring, summer, fall, autumn and winter. Spring and fall are the so-called diplomatic phases, where players are able to negotiate cooperation (lines 6-11). DipBlue starts by revising peace agreements (line 7), taking into account what has happened in the previous phases. Friction ratios are updated and peace is broken for those opponents with a ratio above a given threshold. DipBlue will then select the highest power (line 8) as a target, proposing to all opponents currently in $\mathcal{P}$ an alliance to defeat it (line 9). Sets $\mathcal{P}$ and $\mathcal{W}$ are updated according to the responses received. Advisers in $\mathcal{A}$ are then used to evaluate and select move orders to be executed for each of the bot's units (line 10). Finally, for the selected orders support actions are requested from any opponent in $\mathcal{P}$ having a neighboring region (line 11).

Summer and autumn are phases where orders are executed (lines 12-13), and in case of standoffs, losing units need to retreat to an empty neighboring region or removed from the game. DipBlue uses its advisers in $\mathcal{A}$ to decide which retreat orders to execute for each dislodged unit (line 13).

Finally, winter is the phase where players earn additional units or lose exceeding ones according to the number of supply centers they occupy (lines 14-15). Again, DipBlue uses its advisers to decide where to place its newly acquired units or which units to remove (line 15).

After submitting its orders to the game for execution (line 17), DipBlue will analyze every executed order from its opponents (lines 18-24), and update ratios (line 21) for those players that have decided to attack it, i.e., that have executed move actions to one of its controlled supply centers (line 20).

It is important to emphasize that, for the sake of clarity, we have left outside this algorithm DipBlue's behavior in terms of responses to incoming peace, alliance or order requests. This behavior is informally described in Section 4.2.

5 Experimental Evaluation

When testing the performance of DipBlue, we were confronted with the lack of negotiating bots available for the DipGame platform. In order to overcome this problem, DipBlue archetypes have been tested in an incremental fashion. For that, a number of scenarios have been designed, as listed in Table 1.

In each scenario 70 games were made with the same specifications, and average and standard-deviation data has been computed. Following Diplomacy's rules, in each game 7 players are in play, which are randomly assigned to 7 different world powers.

Given the initially deterministic nature of DipBlue's archetypes, the usage of NoPress as the baseline version for measuring the performance of the remaining archetypes brought a significant number of games ending in a tie due to deadlocks. To avoid this, a simple randomization in the evaluation of moves was

Table 1. Testing scenarios.

#	Bots	Purpose
1	1x NoPress 6x DumbBot	Test the baseline version of DipBlue, which is theoretically equivalent to DumbBot
2	1x Slave 6x NoPress	Test the performance of Slave when facing NoPress (no negotiation)
3	1x Naive 6x NoPress	Test the performance of Naive when facing NoPress (no negotiation)
4	1x DipBlue 6x NoPress	Test the performance of DipBlue when facing NoPress (no negotiation)
5	1x Slave 1x Naive 5x NoPress	Test the performance of the Naive archetype in the presence of a Slave, an agent that accepts and follows any proposed and feasible deal
6	1x Slave 1x DipBlue 5x NoPress	Test the performance of DipBlue in the presence of a Slave, an agent that accepts and follows any proposed and feasible deal
7	1x Naive 1x DipBlue 5x NoPress	Test the performance of DipBlue in the presence of a Naive, a deliberative team-player
8	2x DipBlue 5x NoPress	Test the performance of DipBlue when paired with an equal player, which is also able to perform/detect betrayals

added to NoPress (consisting of adding between -5% and 5% to the evaluation obtained by the MapTactician adviser). Tests showed that this randomization had no significant effect on performance in non-tied games, allowing us to nearly eliminate tie occurrences.

5.1 Crude Performance

As far as the rules of Diplomacy are concerned, the performance of a bot is determined by the position in which the bot ends the game. In games made with 7 equal bots, the average position is the 4th place.

By analyzing the average position obtained by NoPress in Scenario 1, which is 4.1, as shown in Figure 3, it is possible to conclude that the performance of the bot is slightly lower than the performance of DumbBot. Because of this handicap and since NoPress is the foundation for all other bots and has no negotiation capabilities, the best way to measure the improvements of the remaining bots is to compare them with NoPress, rather than DumbBot.

In Scenarios 2-4 we test a Slave, a Naive and a DipBlue against 6 NoPress bots. In this setting, bots are not able to take advantage of negotiation; therefore, they rely on the same heuristics as NoPress, with the exception that DipBlue has trust reasoning based on the opponents actions. Both Slave and Naive were expected to perform exactly as NoPress and in fact they end up with a very similar average position. DipBlue, however, was expected to perform better than NoPress, given its ability to update friction ratios. It was actually capable of achieving a score of 3.61.

Scenarios 5 is used to assess how Naive and Slave work together, since Slave never rejects requests and Naive does not perform trust reasoning, this scenario should act as a reference point to assess the impact of a Slave when paired with

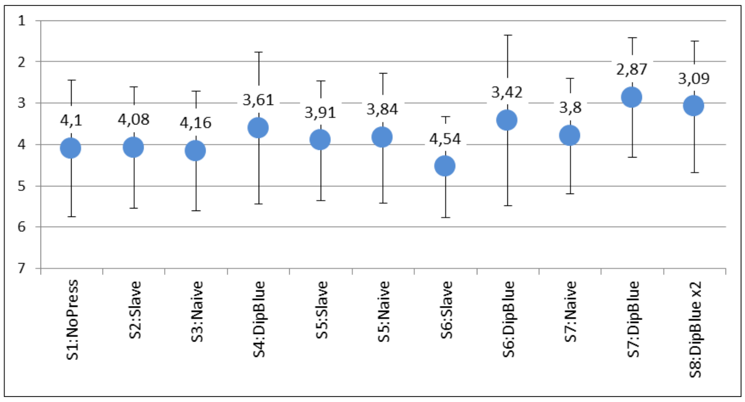

Fig. 3. Average and standard deviation of the final position of DipBlue archetypes in each scenario.

other players, since a Slave may behave like a support player when allied to a player with the proper negotiation capabilities; Slave can be seen as a lever to other players and not as the subject of study itself. Although both bots have a slight performance increase, it is not significant to conclude that the communication between these bots was actually beneficial.

Scenarios 6 and 7 are used to evaluate how DipBlue interacts with communicating bots that do not perform trust reasoning. In Scenario 7, the Naive bot is able to accept a request from DipBlue and then decide not to follow the agreement. This will trigger DipBlue's trust reasoning capabilities and affect the way allies engage in future iterations. Results from this scenario show an obvious increase of performance for both bots, meaning that although they are able to betray each other, they are also able to successfully use the alliance in order to achieve mutual benefit, even more than when each of these bots is paired with a Slave (in Scenarios 5-6). From Scenarios 6-7 it also follows that DipBlue is able to get better results the more challenging its opponent archetype is. DipBlue's highest average position was obtained in Scenario 7: 2.87.

For Scenario 8, Figure 3 shows the average position of both DipBlues. When paired with another instance of itself, DipBlue is able to detect betrayals and is vulnerable to be detected betraying. Therefore, when two instances of this bot are matched there is a high probability of conflict between two former allies. The results highlighted by this scenario display an increase of performance when compared to Scenario 4 (with no negotiation taking place) and also when compared to Scenario 6 (where DipBlue was able to take advantage of a Slave bot). However, there is a decrease when compared to the performance of DipBlue in Scenario 7. While in that scenario DipBlue was able to betray alliances without repercussions, in Scenario 8 betrayals can be detected, which leads to a decrease of performance for the two bots.

From these observed results, we can state that negotiation capabilities allow DipBlue to enhance its results: all negotiation-able archetypes have shown a better performance than NoPress, except when used as a lever to other players, as shown with the Slave in Scenario 6. We can also conclude that mutual negotiation, where each player proposes agreements instead of just one player proposing and the other one following, works better than a "master/slave" scenario.

Furthermore, the implemented model for trust reasoning is an asset when in the presence of adversaries that are capable of negotiating and establishing agreements. Scenario 8 shows how both DipBlue's are able to achieve a good score while avoiding being exploited by the other one (something Slave and Naive are not able to do, in Scenarios 6 and 7, respectively).

Given the final goal of Diplomacy, which is to win the game, we also analyzed the winning capability of DipBlue. Figure 4 shows the percentage of wins of DipBlue in Scenarios 4, 6 and 7. Even though DipBlue has a better average performance with Naive in Scenario 7 (see Figure 3), it is able to win more often by taking advantage of a Slave opponent. In fact, it is able to make slightly more wins when playing alone (as far as negotiation is concerned) in Scenario 4 than when paired with Naive, a much more challenging opponent than NoPress.

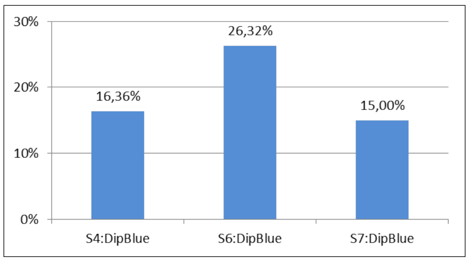

Fig. 4. Win percentage of DipBlue.

5.2 Correlation of Variables

In order to deepen the analysis of the obtained results, we have performed an inspection of dependencies between different variables, by defining correlation coefficients. All correlation coefficients regard the player position, which represents the ranking of the player. Therefore, negative coefficients mean the bigger the value of the variable the better the player's rank.

Figure 5 displays the inverse correlation coefficients using aggregated data from all scenarios. Variables represent: number of years the game takes to end, distance to allies and enemies, percentage of moves cut (i.e. moves invalidated by other player moves), and the number of holds, moves and supports.

The correlation of the final position with the year in which games end is very reduced, meaning there is not a significant dependency between the length of the game and the performance of bots. The same applies to the percentage of moves that have been cut.

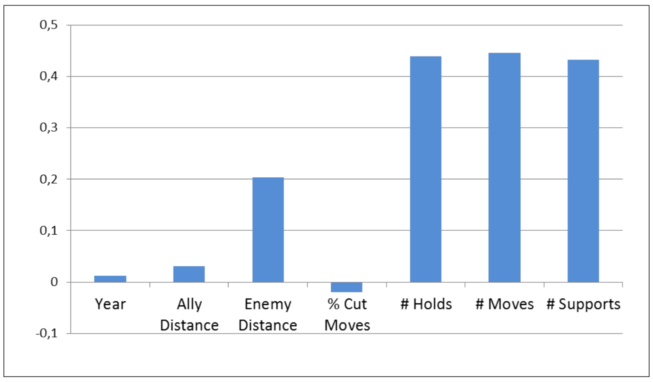

Fig. 5. Inverse correlation with final position of the bot.

The correlation of the distance to allies has a low positive value, which indicates that a slight tendency of a gain in performance is obtained with the increase of the distance. However, it is not significant. Regarding the distance to enemies, Figure 5 shows a significant correlation, which indicates that the farther the enemies are from the player, the better its performance.

Regarding the number of holds, moves and supports, these values display a high correlation with the final position. This is explained by the fact that by having more supply centers, a winning player also has a higher number of units, which in turn implies that it will be executing more actions. Therefore, the number of actions has a direct impact on the position of the player. The high correlation of the number of supports also indicates that teamwork is a good strategy, both when using the player's own units or recurring to those of its allies.

5.3 Impact of World Power

In Diplomacy, world powers determine to some extent the players' performance. One study made by Hunter [7] shows, for games played by humans, a discrepancy between powers both in no-press and in full-press games, as shown in Figure 6. The results show an advantage of nearly double win percentage between France and Italy in both cases. Furthermore, one interesting result is that negotiation seems to only benefit Russia – in every other case performance seems to be better in the no-press variant of the game.

While bot capabilities are far from being comparable to the performance of humans, it is an interesting exercise to compare power discrepancies when it comes to DipBlue's archetypes. Figure 7 shows such a comparison between NoPress (in 140 games ran with 7 NoPress bots) and the results obtained by DipBlue in Scenarios 4, 6 and 7. As shown, DipBlue seems to be able to enhance its performance in almost every case. As in human games, DipBlue achieves its

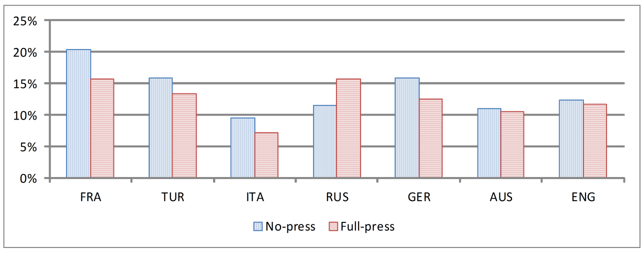

Fig. 6. Win percentages according to power in human games.

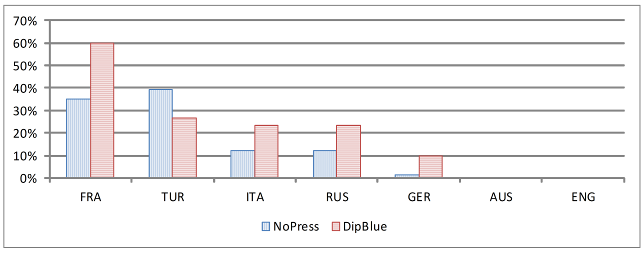

Fig. 7. Win percentages according to power in bot games.

best performance when playing France. Unlike human games, though, neither NoPress nor DipBlue are able to win any game when playing Austria or England. This is partially explained by the lack of convoys in DipGame, which makes it really hard for England to move through the Diplomacy board.

6 Conclusions and Future Work

Addressing multi-player games with cooperative strategies is a challenging domain for multi-agent systems. In this paper we have put forward an initial approach to develop negotiation-based agents for playing Diplomacy. The proposed modular architecture for DipBlue allowed us to test our bot using several different archetypes. The test scenarios had the purpose of highlighting certain aspects of the bots or their combination, producing results that allow to verify the validity of the proposed approach.

As a summary, we conclude that the proposed approach, DipBlue, successfully takes advantage of negotiation, as an alternative (or complement) to traditional solution search approaches. The lack of DipGame bots that are able to enter into negotiations has prevented us from a deeper analysis of our bots virtues. Nevertheless, we may say that negotiation is proven to be a very powerful approach in games where (temporary) cooperation between the players can take place. Furthermore, trust reasoning is a promising direction to address the breaking of agreements.

In the near future we would like to build, using DipBlue's architecture, different deliberative strategies for the game, better exploring negotiation features. This will also allow us to enrich our experiments by populating them with different negotiating bots for DipGame. Consequently, it will also enable us to make a deeper evaluation of the competitive advantages of each strategy as compared to the others.

Some promising improvements to DipBlue are planed, along the following lines.

Performance of World Powers. Bots performance varies greatly according to the world power they are assigned to. Reducing this effect would be beneficial to achieve a more stable and robust player, capable of having a good performance regardless of the world power assigned to it. However, as we have pointed out in our evaluations this is not an easy task, even for human experts.

Communication Capabilities. Negotiation strategies rely on communication. One of the most valuable improvements to be made is to increase the communication capabilities of the bot towards higher levels of the L Language.

Trust Reasoning. DipBlue performs a very simplistic trust reasoning. Being able to combine the previous actions of players with the current state of the game should enable a better assessment of the odds related with establishing or breaking agreements. Opponent modeling techniques and better game state evaluation functions should be used in order to better assess which are the expected strategic moves of opponents in each stage of the game.

Optimization. Following the approach described in [9], which applies genetic algorithms to optimize DumbBot [16], it should be possible to determine the best configuration of DipBlue, in order to achieve an optimal bot.

Learning. Using machine learning techniques, the bot can be endowed with the ability to learn from its previous experiences and opponents after a fair amount of games. This could be used to learn when to play each available action during the game or to improve negotiation tactics. Learning could also be used to predict the next opponent moves.

Finally, it is worth mentioning that the recent growth in MAS research applied to Diplomacy has given rise to a Computer Diplomacy Challenge in the Computer Olympiad 2015 event.

References

1. Allan B. Calhamer. *The Rules of Diplomacy.* Avalon Hill, 4th edition, 2000.
2. Rui Jorge Gregório Deyllot. *Diplomacy Base de Dados de Movimentos para Controlar Províncias.* Master thesis, Universidade de Aveiro, 2010.
3. Angela Fabregues and Carles Sierra. Diplomacy game: the test bed. *PerAda Magazine, towards persuasive adaptation*, pages 5–6, 2009.
4. Angela Fabregues and Carles Sierra. A testbed for multiagent systems (technical report iiia-tr-2009-09). Technical report, IIIA-CSIC, 2009.
5. Angela Fabregues and Carles Sierra. Dipgame: A challenging negotiation testbed. *Engineering Applications of Artificial Intelligence*, 24(7):1137–1146, 2011.

6. Michael R. Hall and Daniel E. Loeb. Thoughts on programming a diplomat. In H. van den Herik and V. Allis, editors, *Heuristic Programming in Artificial Intelligence 3*, pages 123–145. Chinester, England, Ellis Horwood Limited, 1992.
7. Eric Hunter. Solo percentages. `http://www.diplom.org/Zine/W2003A/Hunter/Solo-Percentages.html`. Accessed: 29-07-2015.
8. Stefan J. Johansson and Fredrik Håå rd. Tactical coordination in no-press diplomacy. *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems - AAMAS '05*, page 423, 2005.
9. Dave De Jonge. *Optimizing a Diplomacy Bot Using Genetic Algorithms*. Master thesis, Universitat Autònoma de Barcelona, 2010.
10. Markus Kemmerling, Niels Ackermann, Nicola Beume, Mike Preuss, Sebastian Uellenbeck, and Wolfgang Walz. Is human-like and well playing contradictory for diplomacy bots? In *Proceedings of the 5th International Conference on Computational Intelligence and Games*, CIG'09, pages 209–216, Piscataway, NJ, USA, 2009. IEEE Press.
11. Markus Kemmerling, Niels Ackermann, and Mike Preuss. Nested look-ahead evolutionary algorithm based planning for a believable diplomacy bot. *Applications of Evolutionary Computation*, 6624:83–92, 2011.
12. Sarit Kraus and Daniel Lehmann. Diplomat, an agent in a multi agent environment: An overview. Technical report, Leibniz Center for Research in Computer Science, 1987.
13. Sarit Kraus, Daniel Lehmann, and Eithan Ephrati. An automated diplomacy player. In D. Levy and D. Beal, editors, *Heuristic Programming in Artificial Intelligence: The 1st Computer Olympiad*, pages 136–153. Ellis Horwood Limited, Chinester, UK, 1989.
14. Aleksander Krzywinski, Weiqin Chen, and Arne Helgesen. Agent architecture in social games – the implementation of subsumption architecture in diplomacy. In *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference*, pages 191–196, 2008.
15. Danny Loeb. Challenges in multi-player gaming by computers: A treatise on the diplomacy programming project. `http://diplom.org/Zine/S1995M/Loeb/Project.html`. Accessed: 29-07-2015.
16. David Norman. Dumbbot algorithm. `http://www.daide.org.uk/index.php?title=DumbBot_Algorithm`. Accessed: 29-07-2015.
17. Sylwia Polberg, Marcin Paprzycki, and Maria Ganzha. Developing intelligent bots for the diplomacy game. In *Proceedings of Federated Conference on Computer Science and Information Systems (FedCSIS 2011)*, pages 589–596, 2011.
18. João Ribeiro, Pedro Mariano, and Luís Seabra Lopes. Darkblade: A program that plays diplomacy. In Luís Seabra Lopes, Nuno Lau, Pedro Mariano, and Luís M. Rocha, editors, *Progress in Artificial Intelligence*, volume 5816 of *Lecture Notes in Computer Science*, pages 485–496. Springer Berlin Heidelberg, 2009.
19. S. Russell and P. Norvig. *Artificial intelligence: a modern approach.* Prentice Hall, 3rd edition, 2009.
20. Ari Shapiro, Gil Fuchs, and Robert Levinson. Learning a game strategy using pattern-weights and self-play. In Jonathan Schaeffer, Martin Mller, and Yngvi Bjrnsson, editors, *Computers and Games*, volume 2883 of *Lecture Notes in Computer Science*, pages 42–60. Springer Berlin Heidelberg, 2003.
21. Jason van Hal. Diplomacy ai - albert. `https://sites.google.com/site/diplomacyai/home`. Accessed: 29-07-2015.
22. Adam Webb, Jason Chin, and Thomas Wilkins. Automated negotiation in the game of diplomacy. Technical report, Imperial College London, 2008.