DipBlue: A Diplomacy Agent with Strategic and Trust Reasoning

André Ferreira[1], Henrique Lopes Cardoso[1,2] and Luís Paulo Reis[2,3]
[1]*DEI/FEUP, Faculdade de Engenharia, Universidade do Porto, Porto, Portugal*
[2]*LIACC – Laboratrio de Inteligência Artificial e Ciência de Computadores, Porto, Portugal*
[3]*DSI/EEUM – Escola de Engenharia da Universidade do Minho, Guimarães, Portugal*

Keywords: Diplomacy, Strategy, Negotiation, Trust.

Abstract: Diplomacy is a multi-player strategic and zero-sum board game, free of random factors, and allowing negotiation among players. The majority of existing artificial players (bots) for Diplomacy do not exploit the strategic opportunities enabled by negotiation, instead trying to decide their moves through solution search and the use of complex heuristics. We present *DipBlue*, an approach to the development of 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 multi-agent systems 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.

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. Most approaches to game playing, however, have been based mainly on (adversarial) search techniques and sophisticated domain-specific heuristics. Complex adversarial multi-player games pose new challenges to multi-agent systems (MAS) research: multi-player games with search spaces big enough to render ineffective any approach based solely on search.

Diplomacy is a military strategy multi-player simultaneous move board game, created by Allan B. Calhamer (Calhamer, 2000) and distributed by Hasbro since 1954. Its most interesting attributes include, according to (Hall and Loeb, 1995), the enormous size of its search tree, the difficulty of determining the true strength of a position, and negotiation, whose support brings a competitive advantage to develop sophisticated players.

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 through argumentation.

This work proposes an approach to the creation of an artificial player that takes advantage of negotiation and trust in order to increase its performance. The main goal is to *develop a bot capable of surpassing its opponents by the use of negotiation and trust reasoning*. Our bot, *DipBlue*, works with the MAS testbed *DipGame* (Fabregues and Sierra, 2009) and has been tested with another player of the same platform and with variations of itself.

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 presents an experimental evaluation of DipBlue, and puts forward a set of Diplomacy-related hypotheses on the expected results. Section 6 presents and discusses the obtained results. In Section 7 we draw conclusions of the work done, and we point out directions for future work.

Ferreira A., Lopes Cardoso H. and Reis L..
DipBlue: A Diplomacy Agent with Strategic and Trust Reasoning.
DOI: 10.5220/0005205400540065
In *Proceedings of the International Conference on Agents and Artificial Intelligence* (ICAART-2015), pages 54-65
ISBN: 978-989-758-073-4

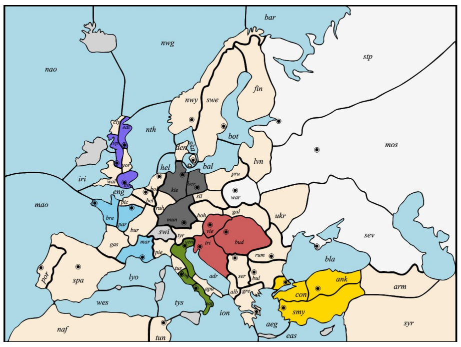

Figure 1: Standard Diplomacy map of Europe

2 DIPLOMACY: THE GAME

Diplomacy takes place in the turn 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 adjacent regions (also termed *attack*), 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 the 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[1].

Before each round of orders, the players are able to communicate willingly with each other, enduring only the restrictions they set among and for themselves. In the negotiation phase of the game players can communicate with each other both publicly and privately. Although these conversations and arrangements are a huge part of the game-play, they hold absolutely no real power in the game itself: a player can commit to execute an action in exchange of information and, after acquiring it, decide not to fulfill its part of the agreement.

Diplomacy is characterized by having no random factors (besides the initial assignment of world powers to players) and being a zero-sum game. However, the size of the game's search tree is enormous and impossible to search systematically even at low depths. To address this problem, in most games the tree is pruned using heuristics that assess the state of the game in 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 (Russell et al., 1995), 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.

In common solution search problems, the perfect or optimal solution is given in a certain depth of the tree and the algorithm proceeds to making the decisions that lead to the optimal solution found. However, when applied to adversary games the solution tree is built with alternate layers of decisions made by the player and decisions made by the opponents. Therefore, the player does not have full control of the course of the game. To deal with the search of the solution space several algorithms were developed over the years, such as Branch and Bound and A*. Since adversary games have a particular kind of search tree, specific algorithms were created to deal with the layered tree, one of the most well-known being Minimax. However, one of the most important aspects of search algorithms is the heuristics used to assess game states. According to some attempts at creating heuristics for Diplomacy, 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.

This 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

A short description of testbeds for Diplomacy and bots developed specifically for this game is provided here. 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.

[1]Detailed rules of the game can be found in (Calhamer, 2000).

The Diplomacy Artificial Intelligence Development Environment (DAIDE[2]) (DAIDE, 2013) assists the development of Diplomacy bots by taking care of all the logic concerning moves validation and the generation of the new 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 simpler or more complex negotiator bots. The communication layers are referred to as *Press levels* and there are 15 distinct ones, ranging from most basic (no communication at all) to the more complex level, able to negotiate in free text. Both server and bots are written in C/C++.

DipGame[3] (Fabregues and Sierra, 2009) is a testbed created at IIIA-CSIC that uses the DAIDE server to handle moves resolution and generation of new game states. Although DAIDE already supports a communication server and its own syntax, DipGame introduces its own server and creates a new communication syntax known as L Language (see Figure 2). 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.

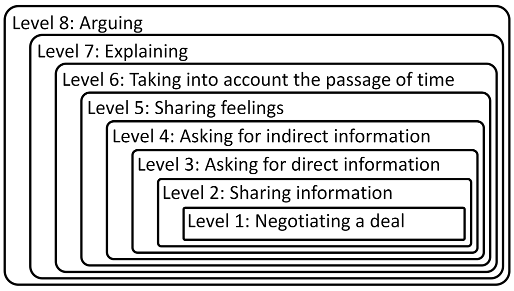
Level 8: Arguing
Level 7: Explaining
Level 6: Taking into account the passage of time
Level 5: Sharing feelings
Level 4: Asking for indirect information
Level 3: Asking for direct information
Level 2: Sharing information
Level 1: Negotiating a deal

Figure 2: Layers of the L Language of DipGame (adapted from (Fabregues and Sierra, 2009)).

3.2 Diplomacy Bots

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

Israeli Diplomat (Sarit Kraus, 1987) was developed in 1988 by Kraus *et al.* to work with a proprietary testbed. 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. The bot has several well designed strategies to deal with solution search and negotiation with opponents.

The Bordeaux Diplomat (Hall and Loeb, 1995) 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 (Norman, 2013) 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 (Jonge, 2010), different only on the lack of support for a move called Convoy, which is not available in DipGame.

The Albert (van Hal, 2013) bot was developed by Jason 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 out-perform it.

BlabBot was created by John Newbury (Webb et al., 2008) and builds on DumbBot by implementing negotiation on top of 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 (Ribeiro et al., 2009) is a no-press bot built by João Ribeiro, which tries to combine the best tactics and strategies used by other Diplomacy agents. DarkBlade follows a modular architecture similar to Israeli Diplomat (see below), and is modeled as an internal MAS, using so-called sub-agents.

HaAI (Johansson and Håå rd, 2005) was developed by Håård and Johansson. It 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 (Polberg et al., 2011) is a DipGame bot developed by Polberg *et al.* and is an extension to the SillyBot, a bot similar to DumbBot (without communication capabilities). SillyNegoBot adds L Language Level 1 communication and includes a

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

[3]http://www.dipgame.org/

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 (Jonge, 2010), 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 (Deyllot, 2010).

3.3 Strategies for Diplomacy

Evaluating board positions is crucial for effective Diplomacy playing. However, as explained before, 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.

The *province destination value* is used by DumbBot to assign a value to each region (Jonge, 2010). 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 on shaky ground 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, like small hints 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 (Webb et al., 2008). 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 stronger player available.

Back-stab is a tactic used by BlabBot for deciding when to betray alliances or for guessing when these will be betrayed by adversaries (Webb et al., 2008). 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 an approach to determine what world powers the player should ask for alliances and which ones to keep the longest. The 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

DipBlue[4] is an artificial player for Diplomacy 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 for the game, a different approach was adopted to tentatively implement an effective Diplomacy bot: DipBlue uses negotiation as its main tool to gain advantage over its competitors, and applies trust reasoning to understand and react when betrayed.

4.1 Architecture

The architecture developed to implement DipBlue has the purpose of being flexible and easily extendible through the use of a highly modular approach, which evaluates and determines the set of orders in each turn from different perspectives. Figure 3 shows a class diagram comprising an overview of DipBlue's architecture, including two main components: Negotiator and Adviser (further explained in Section 4.3). Different advisers may be added as needed to the bot, enabling 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 3 also shows the relation between one of the advisers and DumbBot, the bot it is based on. In

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

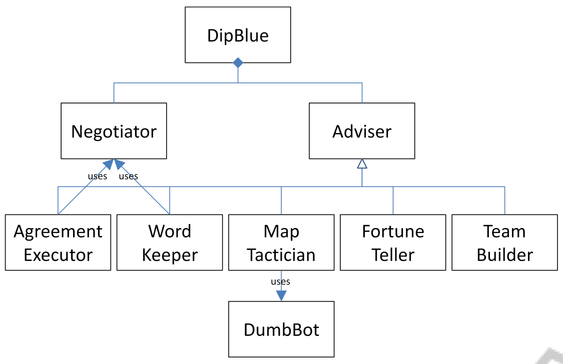

Figure 3: DipBlue architecture.

other terms, DubmBot 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 3).

4.2 Negotiation and Trust

DipBlue is a negotiating bot with the ability to communicate in L Language level 1 (see Figure 2), whose format is explained in (Fabregues and Sierra, 2009). This layer of the language allows for three types of requests: *peace*, *alliance* and *order requests*.

Peace requests reflect the intention for truce to occur among players and it can be understood as a request for cease-fire or simply to achieve neutrality. In an attempt to reduce the probability of conflict with the most players 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 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 requests from other players if they are in a state of peace and if the targeted enemy is not an ally itself.

An order request contains an order regarding a unit of the player to whom the request is sent. It has the purpose of suggesting the other player orders for its 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.

Orthogonal to the use of this negotiation strategy 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 odds on the fulfillment of 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 emphasize the role of betrayals during the game, since this way 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* is associated with each opponent that also reflects the current relationship. 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 to 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 of them is individual and 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 advisers are taken into account.

A ranking of possible orders for each unit is created. The method used to calculate the value assigned to each action is 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 $Order$).

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

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 execution of advisers 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, nulling out 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 (Jonge, 2010).

We now provide short descriptions of the advisers illustrated in Figure 3.

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 if one action in a given set is going to be successful is not a trivial task. Given the size 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, thus often obtaining 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 forfeit its highest ranked action to support 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. Changing 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 with the deal 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 the 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, some different bots were created 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 defined to overcome the lack of DipGame bots available for testing purposes. With the exception of NoPress, every other archetype described below uses all the advisers presented in Section 4.3.

NoPress is the most basic version of DipBlue. It does not perform negotiation of any kind and is unable to perform trust reasoning. It is very similar to DumbBot in terms of capabilities. Advisers: MapTactician, FortuneTeller, TeamBuilder.

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 inflated, 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 trust ratios and using 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. 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 ration 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

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:         $O \leftarrow selectMoveOrders(gameState, \mathcal{A})$
11:         $requestSupports(O, \mathcal{P})$
12:     case $Summer, Autumn$:
13:         $O \leftarrow selectRetreatOrders(gameState, \mathcal{A})$
14:     case $Winter$:
15:         $O \leftarrow selectBuildOrRemoveOrders(gameState, \mathcal{A})$
16:     end switch
17:     $executeOrders(gameState, O)$
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
```

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.

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, al-

Table 1: Testing scenarios.

Scenario	Configuration	Purpose
1	1x NoPress 6x DumbBot	Test the baseline version of DipBlue, which is theoretically equivalent to DumbBot
2	1x DipBlue 6x DumbBot	Test the performance of DipBlue when facing DumbBots, without players to negotiate with
3	1x Slave 1x Naive 5x DumbBot	Test the performance of the Naive archetype in the presence of a Slave
4	1x Slave 1x DipBlue 5x DumbBot	Test the performance of DipBlue in the presence of a Slave, an agent that accepts and follows any proposed and feasible deal
5	1x Naive 1x DipBlue 5x DumbBot	Test the performance of DipBlue in the presence of a Naive, a deliberative team-player
6	2x DipBlue 5x DumbBot	Test the performance of DipBlue when paired with an equal player, which is also able to detect betrayals
7	7x NoPress	Test the baseline version of DipBlue without DumbBots' influence
8	7x DipBlue	Test DipBlue without DumbBots' influence
9	2x DipBlue 5x NoPress	Test the performance of DipBlue when paired with an equal player, without DubmBots' influence

liance or order requests. This behavior is informally described in Section 4.2.

5 EXPERIMENTS

To test the performance of DipBlue archetypes, a number of scenarios have been created, as listed in Table 1. In each scenario 70 games were made with the same specifications, and average 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.

In addition to these scenarios, and in order to better understand the strategic advantage of DipBlue, a number of hypotheses have been formulated, namely:

H1: Close distance allies bring a better performance than long distance ones, as adjacent allies provide lesser contact with enemies and are able to support each other actions.

H2: Being in war with farther opponents is better than with closer ones, as the bigger the distance the less opportunities there are for attacks.

H3: Negotiation is a competitive advantage in Diplomacy, as it endows the player with the ability to temporarily team up with other players.

H4: Trust reasoning increases the performance of the player, as it is able to determine betrayals or aggressive attitudes from its opponents.

H5: Being caught betraying is worst than not being caught, as previous allies may retaliate.

6 RESULTS

After collecting results from several games in each of the scenarios, we have analyzed the results in order to extract useful information.

6.1 Overall Performance

The most relevant result is the position in which the bot ends the game, since it provides a direct insight to the bot's performance. In games made with 7 DumbBots, the average position is the 4th place – since all players have equal performance, there is an even distribution of wins.

By analyzing the average position obtained by NoPress in Scenario 1, which, as shown in Figure 4, is 4.3, it is possible to conclude that the performance of the bot is 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. From this point forward, all references to gain or loss in performance are relative to the values achieved by NoPress.

In Scenario 2, DipBlue faces 6 DumbBots, being therefore unable to take advantage of negotiation, and uses the same heuristics as NoPress plus trust reasoning based on the opponents actions. DipBlue was ex-

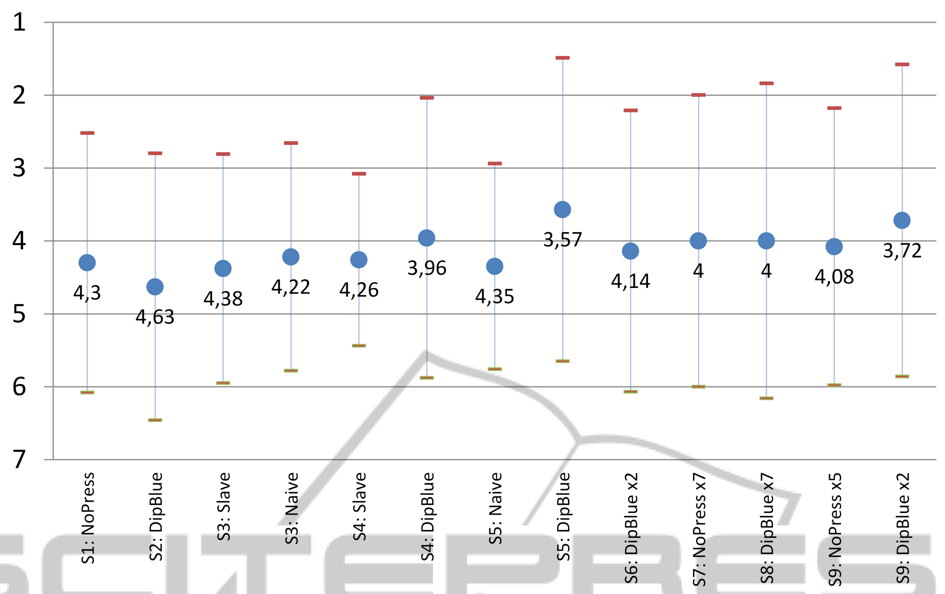

Figure 4: Average and standard deviation of the final position of the bot in each scenario.

pected to perform better than NoPress, given that it has more capabilities. However, it actually performs worst than NoPress, decreasing the average position to 4.63. Since the only difference between both bots, in this particular scenario, is the addition of trust reasoning, a possible reason is that changing trust ratios based on attacks of nearby opponents may not be an optimal strategy since it will increase aggressiveness towards those opponents. In fact, as analyzed in Section 6.2, the player performs better the farthest its enemies are.

Scenarios 3 and 4 are used to assess how Naive and DipBlue act when in the presence of a Slave. A Slave may behave like a support player when allied to a player with the proper negotiation capabilities; the Slave can be seen as a lever to other players and not as the subject of study itself. When paired with Naive, Slave loses performance; however, the Naive bot has a slight increase in performance, which demonstrates the ability to make use of another player for personal benefit, through the use of negotiation. Furthermore, when paired with DipBlue, Slave gains performance and DipBlue displays an advantage over both NoPress and Naive. This indicates that both Naive and DipBlue are able to perform better when in the presence of a Slave and it also shows that DipBlue is capable of a better performance than Naive, due to its trust reasoning.

In Scenarios 5 and 6, DipBlue is paired with a Naive and another DipBlue, respectively, to measure the impact of betrayals and the way DipBlue detects and reacts to them. In Scenario 5, Naive has worst performance than NoPress, than Naive in Scenario 3 and even than Slave when it was also paired with DipBlue in Scenario 4. On the other hand, in Scenario 5 DipBlue achieves the highest score from all tested scenarios, due to its ability to betray the Naive bot and the inability of the latter to detect or react to the betrayal. The results of Scenario 5 are ideal to demonstrate the need for trust reasoning when in the presence of possible betrayals, while it also illustrates the advantages of a player being able to betray its allies.

For Scenario 6, Figure 4 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 Naive in Scenario 3 and NoPress in Scenario 1; however, there is a decrease when compared to the performance of DipBlue in Scenario 5. While in that scenario DipBlue was able to betray alliances without repercussions, in Scenario 6 betrayals can be detected, which leads to a decrease of performance of both bots.

To evaluate how the bots behave when matched against each other without the interference of DumbBots, Scenarios 7 and 8 have been created. Since in both scenarios the performance of every bot was being tracked and all bots were equal to each other, the average position is 4, similarly to when 7 DumbBots are matched. Further analysis of these scenarios revealed that when 7 instances of the same bot are matched against each other, the outcome is the same in all games and the performance of a single instance is determined by the world power the bot is assigned to. The reason why this happens with NoPress and DipBlue but not with DumbBot is because DumbBot has some randomness in the process of choosing the actions, while NoPress and DipBlue have not. Therefore, in each game the player associated with a specific world power will have the same behavior as any equal player that has occupied that world power in a previous game.

Scenario 9 is similar to Scenario 6 in the sense that it matches 2 DipBlues with 5 non-negotiating bots, with the difference of using NoPress as the non-negotiation bot instead of using DumbBot. As expected, DipBlue performs better than NoPress and better than DipBlue in Scenario 6. Since all bots in this scenario lack random factors, the outcome is also dependent on the world powers both DipBlues represent, similar to what happens in Scenarios 7 and 8.

6.2 Correlation of Variables

In order to deepen the analysis of the obtained results, an inspection of dependencies between variables is needed. In order to better understand this dependency a correlation coefficient between variables has been calculated. 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. Given that correlation coefficients proved to be mostly negative, all values were inverted for better understanding and display purposes.

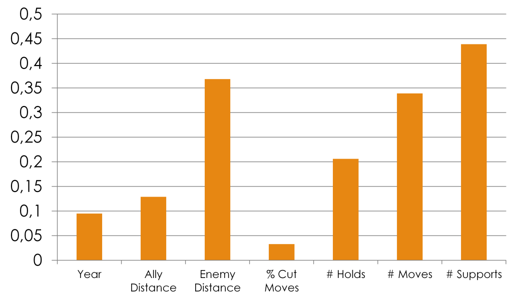

Figure 5: Inverse correlation with final position of the bot.

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

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, explained by the fact that when a player owns several units, these units will in turn perform several actions. Additionally, having several units is correlated with having a higher number of supply centers, which means the player is likely to win the game. Therefore, the number of actions has a direct impact on the position of the player.

6.3 Impact of World Power

In games played by humans there is a slight difference in performance depending on the power the players are assigned to. One study made by Eric Hunter (Hunter, 2014) shows a discrepancy between the powers in tournament games played by humans, which is illustrated in Figure 6. The results show an advantage of nearly double win percentage between France and Italy.

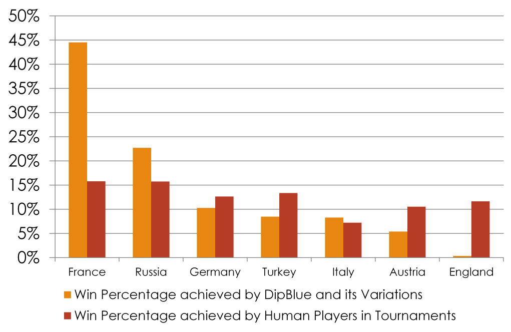

Figure 6: Average position of the bot for each power.

To better understand how the bots behave, the same analysis was made and its results are presented in Figure 6 in comparison with the values obtained by Eric Hunter. Bots display a higher disparity of values as compared to the human tournament results. While the human tournament win percentage ranges from 7.23% to 15.8%, bots win percentages ranged from 0.36% to 44.5%, which indicates that the bots have an accentuated difference in performance depending on the world power they are assigned to. Although the difference between the powers is much larger than in human games, the most successful world powers are the same.

6.4 Revision of Hypotheses

We turn our attention to the hypotheses laid out in Section 5. Hypothesis 1, suggesting an alignment between proximity to allies and performance, was rejected by the results shown in Section 6.2.

Similarly, Hypothesis 2 suggested an alignment between distance to enemies and performance. According to Figure 5, the distance to enemies does have a positive impact on the performance of the player, thus validating this hypothesis.

Hypothesis 3, stating that a bot with negotiation capabilities performs better than a bot without them, can be verified by comparing the results of NoPress with the results of bots that are not being used as a support player, such as DipBlue, as discussed in Section 6.1.

Hypothesis 4 pointed out trust reasoning as an asset. Since trust reasoning was implemented with two distinct elements – based on the actions performed by the opponents and based on the messages sent and received – both variations are analyzed. Following the results discussed in Section 6.1, games played against 6 DumbBots point to the rejection of the hypothesis for action-based trust reasoning, given that NoPress has a better performance than DipBlue. As for negotiation-based trust reasoning, DipBlue achieves a better performance when matched with Naive than when matched with another DipBlue, also capable of betraying and detecting betrayals. This result also validates Hypothesis 5, which states that a bot loses performance if its betrayals are detected.

7 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 negotiationable 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.

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.

Optimization. Following the approach described in (Jonge, 2010), which applies genetic algorithms to optimize DumbBot (Norman, 2013), 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.

REFERENCES

Calhamer, A. B. (2000). *The Rules of Diplomacy*. Avalon Hill, 4th edition.

DAIDE (2013). DAIDE Homepage. http://www.daide.

org.uk / w/index.php? title=Main_Page. Accessed: 15-07-2013.

Deyllot, R. J. G. (2010). *Diplomacy Base de Dados de Movimentos para Controlar Províncias*. Master thesis, Universidade de Aveiro.

Fabregues, A. and Sierra, C. (2009). A Testbed for Multiagent Systems (Technical Report IIIA-TR-2009-09). Technical report, IIIA-CSIC.

Hall, M. R. and Loeb, D. E. (1995). Thoughts on Programming a Diplomat. *Heuristic Programming in Artificial Intelligence*.

Hunter, E. (2014). Solo Percentages. http://www.diplom.org/Zine/W2003A/Hunter/Solo-Percentages.html. Accessed: 04-01-2014.

Johansson, S. J. and Håå rd, F. (2005). Tactical coordination in no-press diplomacy. *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems - AAMAS '05*, page 423.

Jonge, D. D. (2010). *Optimizing a Diplomacy Bot Using Genetic Algorithms*. Master thesis, UAB.

Norman, D. (2013). David Norman's DumbBot. http://www.daide.org.uk/w/index.php?title=DumbBot_Algorithm. Accessed: 12-07-2013.

Polberg, S., Paprzyck, M., and Ganzha, M. (2011). Developing intelligent bots for the Diplomacy game. In *Computer Science and Information Systems*, pages 589–596.

Ribeiro, J. a., Mariano, P., and Seabra Lopes, L. (2009). Darkblade: A program that plays diplomacy. In Lopes, L. S., Lau, N., Mariano, P., and Rocha, L. M., editors, *Progress in Artificial Intelligence*, volume 5816 of *Lecture Notes in Computer Science*, pages 485–496. Springer Berlin Heidelberg.

Russell, S., Norvig, P., Canny, J., Malik, J., and Edwards, D. (1995). *Artificial intelligence: a modern approach*. Prentice Hall, 3rd edition.

Sarit Kraus, D. L. (1987). Diplomat, an Agent in a Multi Agent Environment: An Overview. Technical report, Leibniz Center for Research in Computer Science.

van Hal, J. (2013). Jason van Hal's Homepage. https://sites.google.com/site/diplomacyai/ home. Accessed: 15-07-2013.

Webb, A., Chin, J., and Wilkins, T. (2008). Automated negotiation in the game of diplomacy. Technical report, Imperial College London.