FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

DipBlue: a Diplomacy Agent with Strategic and Trust Reasoning

André Filipe da Costa Ferreira

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Henrique Lopes Cardoso

Co-Supervisor: Luís Paulo Reis

January 13, 2014

DipBlue:
a Diplomacy Agent with Strategic and Trust Reasoning

André Filipe da Costa Ferreira

Mestrado Integrado em Engenharia Informática e Computação

January 13, 2014

Abstract

Diplomacy is a military strategy turn-based board game, which takes place in the turn of the 20th century, where seven world powers fight for the dominion of Europe. The game can be played by 2 to 7 players and is characterized by not having random factors, as well as, by being a zero-sum game. It has a very important component when played by human players that has been put aside in games typically addressed by Artificial Intelligence techniques: before making their moves the players can negotiate among themselves and discuss issues such as alliances, move propositions, exchange of information, among others. Keeping in mind that the players act simultaneously and that the number of units and movements is extremely large, the result is a vast game tree impossible of being effectively searched. The majority of existing artificial players for Diplomacy don't make use of the negotiation opportunities the game provides and try to solve the problem through solution search and the use of complex heuristics.

This dissertation proposes an approach to the development of an artificial player named DipBlue, that makes use of negotiation in order to gain advantage over its opponents, through the use of peace treaties, formation of alliances and suggestion of actions to allies. Trust is used as a tool to detect and react to possible betrayals by allied players. DipBlue has a flexible architecture that allows the creation of different variations of the bot, each with a particular configuration and behaviour. The player was built to work with the multi-agent systems testbed DipGame and was tested with other players of the same platform and variations of itself. The results of the experiments show that the use of negotiation increases the performance of the bots involved in the alliances if all of them are trustworthy, however, when betrayed the efficiency of the bots drastically decreases. In this scenario, the ability to perform trust reasoning proved to successfully reduce the impact of betrayals.

Keywords Games, Multi-Agent Systems, Artificial Intelligence, Diplomacy, DipGame, Strategy, Negotiation, Trust

Resumo

O Diplomacy é um jogo de tabuleiro de estratégia militar, de turnos, passado no virar do século vinte onde sete potências lutam pelo domínio da Europa. O jogo é jogado por 2 a 7 elementos e caracteriza-se por não possuir factores aleatórios, bem como, por ser um jogo de soma-zero. Este tem uma componente bastante importante quando jogado entre jogadores humanos e que tem sido descartada nos jogos tipicamente abordados por Inteligência Artificial: antes de efectuarem as jogadas, os jogadores podem negociar entre si e discutir assuntos como alianças, propostas de jogadas, trocas de informações, entre outros. Tendo em conta que os jogadores actuam simultaneamente e que o número de unidades e movimentos é bastante extenso, o resultado é uma árvore de jogo demasiado vasta para ser pesquisada eficazmente. A maioria dos jogadores existentes para Diplomacy não tiram proveito das oportunidades que o jogo proporciona e tentam resolver o problema através de pesquisa de soluções e do uso de heurísticas complexas.

Esta dissertação propõe uma abordagem para a criação de um jogador artificial chamado DipBlue, que tire proveito da negociação de forma a obter vantagem em relação aos restantes jogadores, através do uso tratados de paz, formação de alianças ou sugestão de acções a aliados. É ainda usada confiança como um meio de detectar e reagir a possíveis traições por parte de jogadores aliados. O jogador foi criado para a plataforma de testes de sistemas multi-agente DipGame e foi testado contra outros jogadores da mesma plataforma e contra variações de si mesmo. Os resultados das experiências demonstram que o uso de negociação aumenta a performance dos bots aliados se todos forem fieis aos acordos efectuados, contudo, quando traídos a eficácia dos bots desce drasticamente. Neste cenário, a capacidade de avaliar confiança provou ser capaz de reduzir o impacto das traições.

Palavras Chave Jogos, Sistemas Multi-Agente, Inteligência Artifical, Diplomacy, DipGame, Estratégia, Negociação, Confiança

Acknowledgements

First of all I would like to thank my supervisors Henrique Lopes Cardoso and Luís Paulo Reis for their support and guidance and for their tips and ideas when mine were gone. I would also like to thank Dave de Jonge from IIIA-CSIC who provided precious help and information regarding the platform and implementation of the bot.

I owe a special thanks to my girlfriend for her patience and encouragement and for always pointing out the right way. And to my parents and grandparents who always made everything to help me make this possible.

"Luck plays no part in Diplomacy. Cunning and cleverness honesty and perfectly-timed betrayal are the tools needed to outwit your fellow players. The most skilful negotiator will climb to victory over the backs of both enemies and friends.
Who do you trust?"

Avalon Hill

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Objectives . . . 3
- 1.3 Document's Structure . . . 4

2 Diplomacy Game **7**
- 2.1 Rules of Diplomacy . . . 8
  - 2.1.1 Map . . . 8
  - 2.1.2 Units . . . 9
  - 2.1.3 Game Phases . . . 10
- 2.2 Diplomacy Testbeds . . . 12
  - 2.2.1 DAIDE . . . 13
  - 2.2.2 DipGame . . . 14
- 2.3 Summary . . . 14

3 Related Work **17**
- 3.1 Diplomacy Bots . . . 17
  - 3.1.1 DAIDE Bots . . . 17
  - 3.1.2 DipGame Bots . . . 19
  - 3.1.3 Other Bots . . . 19
- 3.2 Negotiation Strategies . . . 21
  - 3.2.1 Peace to All . . . 21
  - 3.2.2 BackStab . . . 21
  - 3.2.3 Power Cluster . . . 21
- 3.3 Evaluation Heuristics . . . 22
  - 3.3.1 Province Destination Value . . . 22
  - 3.3.2 Blurred Destination Value . . . 22
  - 3.3.3 Learning . . . 22
- 3.4 Solution Search . . . 22
  - 3.4.1 NB^3 . . . 23
  - 3.4.2 Minimax . . . 24
- 3.5 Trust . . . 25
  - 3.5.1 Dynamic Environments . . . 26
  - 3.5.2 Trust Assessment over Time . . . 26
- 3.6 Summary . . . 26

4 DipBlue **29**
4.1 Architecture . . . 30
4.2 Negotiation . . . 31
4.2.1 Trust Ratio . . . 31
4.2.2 Peace . . . 32
4.2.3 Alliances . . . 32
4.2.4 Order Requests . . . 33
4.3 Advisers . . . 33
4.3.1 MapTactician . . . 34
4.3.2 FortuneTeller . . . 35
4.3.3 TeamBuilder . . . 35
4.3.4 AgreementExecutor . . . 35
4.3.5 WordKeeper . . . 36
4.4 Archetypes . . . 36
4.4.1 NoPress . . . 37
4.4.2 Slave . . . 37
4.4.3 Naive . . . 37
4.4.4 DipBlue . . . 38
4.5 Summary . . . 39

5 Experiments and Results **41**
5.1 Set up . . . 41
5.1.1 Game Launcher . . . 41
5.1.2 Player . . . 42
5.1.3 Scout . . . 42
5.2 Scenarios . . . 42
5.3 Results . . . 44
5.3.1 Overall Performance . . . 44
5.3.2 Correlation of Variables . . . 47
5.3.3 Impact of Power . . . 49
5.4 Summary . . . 50

6 Conclusions **51**
6.1 Analysis of Results and Review of Objectives and Hypotheses . . . 51
6.2 Summary and Contributions . . . 52
6.3 Future Work . . . 52
6.3.1 Performance of Powers . . . 53
6.3.2 Communication Capabilities . . . 53
6.3.3 Reputation . . . 53
6.3.4 Optimization . . . 53
6.3.5 Learning . . . 53

References **55**

List of Figures

2.1 Standard Diplomacy map of Europe 9
2.2 An attack from Marseilles with support from Gascony, from [Cal00] 12
2.3 A standoff between Berlin and Warsaw, both units fail, from [Cal00] 12
2.4 Support from Silesia is cut from Bohemia, attack fails, from [Cal00] 12
2.5 Layers of the L Language of DipGame 14

3.1 The architecture of the Israeli Diplomat, from [Rib08] 20
3.2 Example of Blurred Destination Value, before and after the application of blur, respectively. 23

4.1 An overview of the DipBlue 30
4.2 DipGame's Language Level 1. *Predicate* is either peace or alliance, *action* is any order performed by a unit and *agent* is a power. 31

5.1 Average and standard deviation of the final position of the bot in each scenario. . 45
5.2 Inverse correlation with final position of the bot. 48
5.3 Average position of the bot for each power 49

LIST OF FIGURES

List of Tables

2.1 Colour corresponding to each world power in the map shown in Figure 2.1 10
2.2 Press levels of the DAIDE platform 13

4.1 Configuration of NoPress 37
4.2 Configuration of Slave 38
4.3 Configuration of Naive 38
4.4 Configuration of DipBlue 39

5.1 Variables collected by each bot regarding itself, its opponents and the game . . . 43
5.2 Description of test scenarios 44

Abbreviations

AI	Artificial Intelligence
MAS	Multi Agent Systems
DAIDE	Diplomacy Artificial Intelligence Development Environment
IIIA-CSIC	Artificial Intelligence Research Institute of the Spanish Scientific Research Council

Chapter 1

Introduction

One of the main application areas of Artificial Intelligence has been, since its beginning, problem solving, to which search techniques have been developed and refined. One of the fields where solution search has been extensively applied is games, more specifically, to build artificial players that play games. Similarly to other problems addressed through solution search, games have a solution space, typically represented in the form of a graph. The process of searching the tree, which consists in travelling through the positions of the graph, generates the search tree. 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 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 games have a particular search tree, specific algorithms were created to deal with the layered tree, one of the most well-known being Minimax. Minimax has been successfully applied to several games, like Checkers and Chess, by being able to search the entire search tree or by means of heuristics. However, and in addition to having a large solution space, some games have imperfect or hidden information that interfere with the creation of effective heuristics, which is the case of Diplomacy.

This chapter has the purpose of introducing the context of the dissertation and its motivation, along with the proposed objectives and hypothesis. Finally there is a brief overview of the document.

1.1 Motivation

Diplomacy is a military strategy turn-based board game created by Allan B. Calhamer and distributed by Hasbro, since 1954. The game is best played by 7 players but variations of the game exist to support as low as 2 players. It takes place in the turn of the 20th century in the years

before World War I. Each player represents one Country or world power, such as 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 centres throughout the map.

When played by humans, the game's rules are quite easy to follow, there is no need for complicated calculations since one of the main aspects of the game is negotiation between the players. During the game and before each round of moves, the players are able to communicate willingly with each other, enduring only the restrictions they set among and for themselves, time and/or content wise. In the negotiation phase of the game players can communicate with each other both publicly and privately and the subject of these negotiations can range from a simple alliance proposition with a "yes" or "no" answer, to a complicated set of conditions in exchange of valuable information. Although these conversations and arrangements are a huge part of the gameplay, they hold absolutely no real power in the game itself: a player can compromise to execute an action in exchange of information not fulfilling its part of the agreement, after acquiring it.

Diplomacy, as many other board games, is the target of research from areas such as Computer Science, Mathematics and Game Theory. Along with many other games, it provides some challenges concerning the search for the perfect solution or a way to always win, such as the solution found for the game Tic-Tac-Toe. Usually this type of games provide a scenario that is hard to solve by humans due to the large number of calculations and complicated heuristics (take Chess, for example) – these are problems that computers solve easier and faster than humans. Diplomacy is characterized by having no random factors (besides the initial attribution of one player to a world power) and being a zero-sum game (game in which every time a player loses a point, some other player one). These two aspects allow us to narrow the scope of study to games that fit in the same category, such as Checkers and Chess.

However, the size of the Diplomacy game tree is enormous and impossible to search in its entirety even at low depths. To address this problem in other fields of study, the common approach is to prune the tree by using heuristics that assess the state of the game in a given time and compare it to the future game states. This cannot be directly applied to Diplomacy since the game is non-deterministic, therefore the moves of a player do not take effect according to the commands given. Even though a player orders a unit to make a certain action, it may fail by interference of other players' actions, which is explained in detail in Chapter 2. This makes the generation of the future game states a complicated task.

Furthermore, the creation of heuristics that return the value of the game state in a numerical value has been the subject of different experiments over the past few years, from either the scientific community and Diplomacy players, and most of them resulting in the same conclusion: there is no precise way to evaluate the state of the game since the only visible information is the map and one of the major components of Diplomacy is the negotiation among the players. According to some attempts at creating heuristics, a player can be considered a weak opponent because of

the number and placement of his armies, but can have many strong alliances and, with that alone, a player can win the game or annihilate another player in a few turns.

Diplomacy offers an amazing environment for testing negotiation between the players. When played correctly, two players can achieve a success ratio of almost 100%, since both players can work together, coordinating their actions similar to one single player. In addition, each player can individually gain the trust of other players and share critical information with their allies. This rich environment allows the existence of bots capable of dominating their opponents through negotiation which increases the need for trust reasoning capabilities to allow players to protect themselves.

1.2 Objectives

This dissertation proposes an approach to the creation of an artificial player that takes advantage of negotiation and trust in order to increase the overall performance of the player. One of the main goals is to develop mechanisms that are able to take advantage of negotiation to form alliances that persuade other players to work together towards a common goal. Assuming that the opponent maintains its part of the deal and speaks the truth, there is no need for trust reasoning. However, the arrangements might not be fulfilled and the need for trust emerges. By assessing the trust held in each player regarding the deals made and the actions performed, it is possible to better understand the reliability of an ally and the probability of it breaking the deals made.

The main objective of this dissertation is thus to **develop a bot capable of surpassing its opponents by the use of negotiation and trust reasoning**. The bot created will work with the MAS testbed DipGame [Fab13] and will be tested with one other player of the same platform and with variations of itself. To enable negotiation with other players, the bot will make use of L Language which the communication specification used by DipGame.

To achieve this goal, the following objectives must be pursued:

Objective 1 Implement a simple solution search based foundation to allow the bot to function without negotiation and to create a baseline to test the performance of the bots

Objective 2 Implement the ability to communicate according to L Language Level 1 (see Figure 2.5), since it allows the bot to send messages regarding peace requests, alliances formation and action proposals

Objective 3 Create and implement tactics to negotiate with other players and make efficient use of the available communication capabilities

Objective 4 Design an adequate trust model for Diplomacy that allows the bot to detect possible betrayals and assess the benefits of certain deals and alliances

Objective 5 Develop a flexible and scalable architecture in order to easily create variations of the bot and implement new modules

Finally, the following hypotheses were formulated based on the objectives and will be checked according to the obtained results.

Hypothesis 1 Close distance allies bring a better performance than long distance ones, given that if allies are adjacent to each other they have lesser contact with possible enemies and have the advantage of being able to support each other actions and act like one single player

Hypothesis 2 Being in war with farther opponents is better than with closer opponents, since enemy players will attack each other, therefore, the bigger the distance between the players the less opportunities they have to attack

Hypothesis 3 Communication, and more specifically negotiation is a competitive advantage in Diplomacy, since it endows the player with the ability to team up with other players to achieve a common goal

Hypothesis 4 Performing trust reasoning results in an increase in the performance of the player, considering that the player has a mean to determine betrayals or aggressive attitudes from its opponents

Hypothesis 5 Betraying and being caught is worst than betraying and not being caught, since by being caught the player might suffer repercussions from its previous allies and is not capable of further betrayals

Hypothesis 6 The performance of the player is independent of the world power it is initially assigned to, since the bots make no distinction of the world power the player represents

1.3 Document's Structure

This document is divided into six chapters being the present Chapter 1, which focuses on explaining the context and the motivation of this dissertation. In Chapter 2 a detailed explanation of the rules of the Diplomacy game is given, including the map and regions, the units and the possible and legal moves, and an overview of the main Diplomacy MAS testbeds. In Chapter 3 it is presented a study of the existing related work concerning the main bots for Diplomacy and their approaches to the game as well as some approaches to problems that require similar techniques. The proposed solution is presented in Chapter 4, divided in the architecture of the bot, the negotiation tactics, the Advisers and the bot archetypes. All tests and experiments are described in Chapter 5, along with the results of the different test scenarios and a revision of the hypotheses. Finally, in Chapter 6

reside the conclusions of the work and its objectives, the contributions made by this dissertation
2 and possible future improvements.

Chapter 2

Diplomacy Game

This chapter is focused on the Diplomacy game, its rules and some testbeds that implement Diplomacy as a MAS game scenario for recreation and research purposes.

Diplomacy is a military strategy game created by Allan B. Callamer in 1954 an first published in 1959. It is a military strategy turn-based game where seven Countries or world powers fight to conquer Europe, in the turn of the 20th century, right before World War I. The game has earned a lot of attention since the 70's, and there were a lot of attempts to play the game in a different way than the traditional board game, in order to improve the versatility of the communication to ease the setting up of the game and the players. The game has had a huge adhesion in its letter and mail version where players have turns of typically one week. One player takes the role of the game master and collects all the orders and sends back the result of the given week actions. With the advance of technology, similar versions of the letter and mail game appeared in a phone version and in the last years, over the internet. The first electronic version of the game appeared in 1984, published by Avalon Hill, then the rights passed on to Paradox Interactive and finally the game has been published and distributed by Wizards Of the Coast up until now [otC13].

The main difference, and one of the reasons why Diplomacy has been the target of so much attention by researchers in the last years, is the size of its game tree, which even compared to Chess, is preposterous. Many Chess players consider the game to be amazing for the fact that there is no optimal and final solution to the game, in other words the game hasn't been solved to completion. The reason for this lies in the size of its game tree. Chess has an average branching factor of 35 [Sha50], which gives a total of 1.225 openings, just by looking one move ahead ($35^2 = 1225$). However, because of the size the tree reaches with a total average depth of 80 [Sha50], the best Chess artificial player, DeepBlue, only used a depth between 6 and 16, reaching 40 in some critical branches [Cam01]. In Diplomacy, players act in parallel with each other, all moves are made simultaneously, which means that in each round all seven players make an action with all their units. Because of this, Diplomacy has an huge branching factor, since in each turn there are approximately 25 different units, each of which can make an average of 10 separate moves. The

average depth of a Diplomacy game is 20 [Sha13] when played by humans but tends to reach 30 or 40 when played by bots. This scales to the point of having a total of 4.430.690.040.914.420 unique openings [Joh06]. This is what makes the creation of artificial players challenging and requires different approaches, rather than the common tree search solution. Allan Ritchie did a very thorough analysis and comparison of Diplomacy and other board games [Rit03].

2.1 Rules of Diplomacy

This section describes only the essential rules needed for an easy understanding of the game and to ease the reading of some strategies and tactics explained in Chapter 4. For further reading and to better understand the rules of the game read the official Diplomacy rule book by Allan B. Calhamer [Cal00]. There are several different versions of the game each with its own variations of the rules, the map and even the number of players. Throughout this dissertation only the standard version of the game is comprehended, with seven players and the standard map of Europe (see Figure 2.1), with 34 supply centres.

The game is played by seven world powers and in the very beginning of the set up the only random component in the entire game takes place: the choosing of the world power each player will represent. The available world powers are France, England, Germany, Austria, Turkey, Italy and Russia. Every world power starts with 3 armies placed in its original supply centres, with the exception of Russia that starts with 4 – the reason for this can be found in the explanation of the game by the author [Cal13]. This setup phase can be done by any means the players desire, usually the names of the powers are written in papers and shuffled together each player taking one at random. After knowing which world power the player will represent, the map can be assembled by placing the corresponding units in the home country supply centres, which are represented in Figure 2.1. The map represents the initial setting of the game. The main goal of the game is to conquer the major portion of Europe and ends when one player controls at least 18 of the 34 supply centres. When this happens, the player is named the winner and the remaining players are ranked according to the number of supply centres, if they are still playing, and by the year of loss, if they are already defeated.

2.1.1 Map

The map of the game is divided into 75 regions or provinces, 34 of them containing a supply centre shown in Figure 2.1, represented with a black dot inside the region. Initially, each power controls its home supply centres shown in the map as the regions with colors different than beige. Each different color represents a world power, except for the light blue which is the sea; the color corresponding to each power can be seen in Table 2.1. There are 3 types of regions: inland, coastal and sea. Inland regions do not have sea in any of its frontiers, coastal regions are land regions that have sea in at least one of its frontiers and sea regions are areas inside the sea. The neighbours

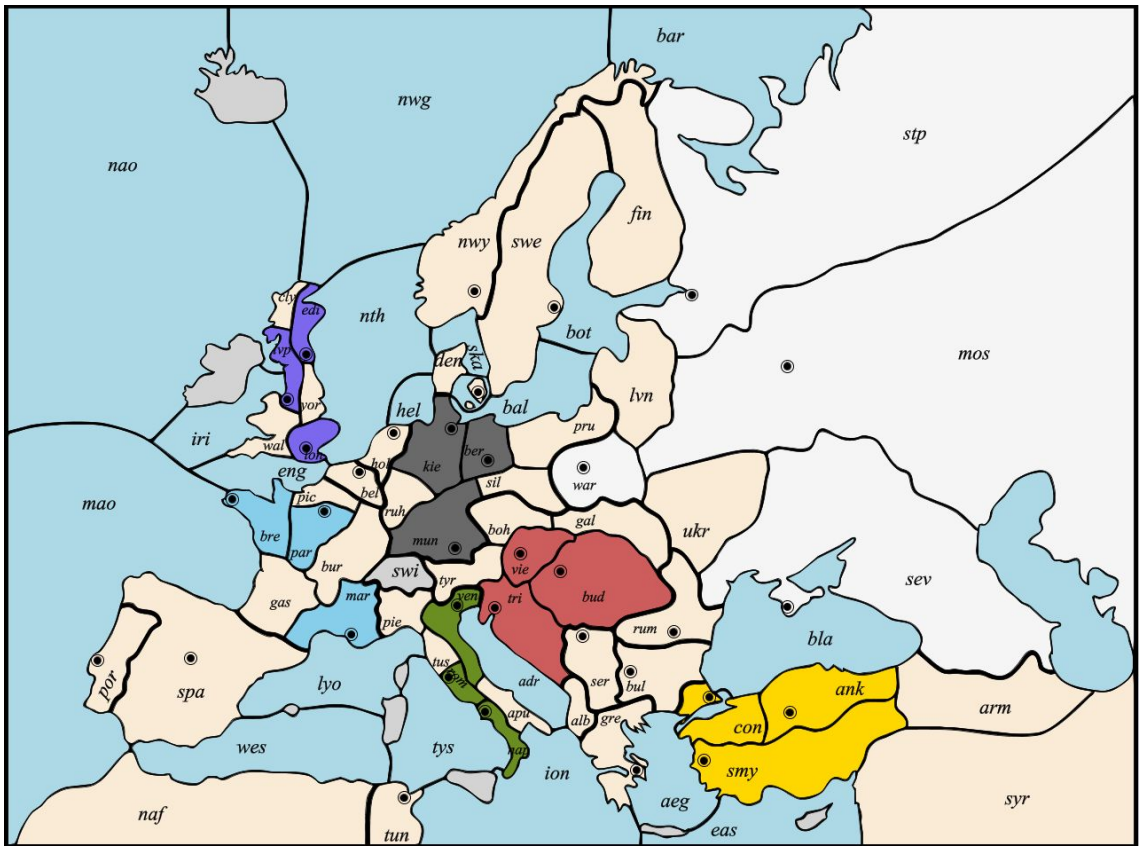

Figure 2.1: Standard Diplomacy map of Europe

of a region are all the regions that have a common frontier. In the map Portugal (POR), has 2 neighbours: Spain (SPA) and Middle Atlantic Ocean (MAO).

2.1.2 Units

There are 2 types of units in the game, armies and fleets, represented by cannons and ships, respectively. Armies represent infantry units and are able to occupy inland and coastal regions. Fleets represent ship or boat units and are able to occupy sea and coastal regions. However, in this case, the fleet can only be in one coast. For example, a fleet that occupies Spain must be in one of Spain's sea frontiers: North with the Middle Atlantic Ocean or South with the Western Mediterranean Sea. In this case, a fleet can only move to regions that are adjacent to the occupied coast. In the case where a fleet is in the south coast of Spain, the possible moves are: Middle Atlantic Ocean and the Western Mediterranean Sea.

All units are able to move one space, this means, every unit can move to one of the region's neighbours, in one turn. In all cases, only one unit can occupy a region – this applies to all regions and all unit types with no exception. Every unit has the same strength, there is no unit stronger or weaker than the other.

Colour	Power
Blue	France
Dark Blue	England
Grey	Germany
Red	Austria
Green	Italy
Yellow	Turkey
White	Russia

Table 2.1: Colour corresponding to each world power in the map shown in Figure 2.1

2.1.3 Game Phases

The turn-based system of the game is represented by the passage of time, where each phase of the game corresponds to a season of the year. There are 5 phases in each year: spring, summer, fall, autumn and winter. The roles of the phases are displayed below:

Spring
: Diplomatic Phase and Order

Summer
: Resolve Conflicts and Disband of Units

Fall
: Diplomatic Phase and Order

Autumn
: Resolve Conflicts and Disband of Units

Winter
: Gaining or Losing Units

At the end of each year, in the Winter phase, every player is allowed the same number of armies as the number of controlled supply centres. A supply center is controlled by a power if one of its units occupy the region by the end of a Summer or Autumn phase. The region continues under that power's control until a unit of other power occupies it by the end of the previously mentioned phases.

The game always starts in the Spring of 1901 with a negotiation phase where players can communicate freely with each other. This phase may or may not have a time limit, in simulated versions there usually is.

2.1.3.1 Negotiation

In the Diplomacy or Negotiation phase, players can communicate with each other freely and both privately and publicly. The contents of the messages can range from an Alliance proposal with a yes/no answer to a complex chain of requests in order to obtain some kind of information. However, none of the agreements are binding and there is no penalty for breaking a pact or promise. After this phase has ended, players may only speak with each other in the next negotiation phase. They are not allowed to speak in any other phase.

2.1.3.2 Orders

The orders are the commands the player gives to its units. In the board version, orders must be written in paper and stored until every player has done the same. Only then, the map is updated with the results of the orders and the resolution of conflicts. The player may only give one command to each unit. If no order is given, the unit just holds. The possible orders a player can give to its units are Hold, Move and Support.

Hold When a unit is commanded to Hold, it does not move. The unit remains in the place it was in the previous turn. This can be used as a defensive technique in order to not leave the region unprotected.

Move The Move action, also known as Attack, tells a unit to move from one region to a valid neighbour. This is used to take an empty region or to invade an occupied one. This action originates most of the conflicts and is the most used action in the game. The decision of what happens with each unit is done in the Summer and Autumn phases. Whenever a unit tries to move to an empty region, it abandons the former region and takes the new one. When a unit moves to an occupied region, it creates a standoff. Standoffs are solved by matching the strength of each opposing force. As mentioned before, each unit, whether Army or Fleet, has the same strength of 1.

In order to increase the strength of an attacker or a defender, players must use a Support. When the opposing sides of a standoff have equal strength, both actions are nullified and both units return to the region they came from. When the defender is stronger than the attacker, the attacker withdraws and returns to the previous region. When the attacker has the advantage, it gains control of the region and the losing unit must Dislodge or Disband (see Section 2.1.3.3).

Support A unit can be ordered to support another unit's action. The support can be given to either Hold or Move actions. When supporting an holding unit, the support is given to the defence of the unit and increases the strength of the defender in 1. When supporting a moving unit, the support is given to the attack of the unit and increases the strength of the attacker by 1, Figure 2.2 shows an example of the latter. Whatever the case, the supporting unit always returns to the origin region. When supporting an attacker, if the attack is successful, only the unit that had the Move action takes the region. If the supporting unit performs a Move instead of a Support, the two units

will clash in a standoff, see example in Figure 2.3. Support can, however, be denied by being cut. Cutting a support consists of attacking the region where the supporting unit is, forcing it to return and act like it was ordered to Hold, see example in Figure 2.4. Cutting support can be used to reduce the attacking force as opposed to increasing the defensive force.

Figure 2.2: An attack from Marseilles with support from Gascony, from [Cal00]

Figure 2.3: A standoff between Berlin and Warsaw, both units fail, from [Cal00]

Figure 2.4: Support from Silesia is cut from Bohemia, attack fails, from [Cal00]

2.1.3.3 Dislodge/Disband

When a standoff occurs and one of the units losses the battle, it is dislodged from that region. The player must then decide where to move the unit in a sort of retreat. The unit must only move to neighbour regions that are empty. If there are no empty regions adjacent, the unit must disband. When a unit disbands it disappears from the map and is lost to the player.

2.1.3.4 Build

At the end of the year, in the Winter phase, every player earns or loses units depending on the number of occupied supply centres at that moment. If a player wins a supply centre in the Summer phase and losses it in the Autumn phase, the supply centre does not count for the number of units. At the beginning of each year, every player is allowed to own a maximum of units equal to the number of controlled supply centres. If the number of units is bigger, the player is forced to disband units until the numbers are even. In the cases where a player has more supply centres than units, whether by gaining supply centres or by loosing units in battle, the player can create new units in the home supply centres, if and only if, the supply centre is one of the starting supply centres and if it is vacant.

2.2 Diplomacy Testbeds

With the increasing popularity of the game in the scientific community due to some aspects of the game that make it a great scenario for MAS testing and experiments, many attempts have been

Level 0	No Press
Level 10	Peace and Alliances
Level 20	Order Proposals
Level 30	Multi-part Offers
Level 40	Sharing out the Supply Centres
Level 50	Nested Multi-part Offers
Level 60	Queries and Insistences
Level 70	Requests for suggestions
Level 80	Accusations
Level 90	Future discussions
Level 100	Conditionals
Level 110	Puppets and Favours
Level 120	Forwarding Press
Level 130	Explanations
Level 8000	Free text

Table 2.2: Press levels of the DAIDE platform

made to create a platform that supports the creating and testing of artificial players or bots. Such platforms appeared in every other research center, each with an unique way of structuring the map, the players and most of all, the communication protocol. In time, one very popular platform began to emerge and took the place of the standard Diplomacy testbed. The platform is DAIDE and is explained in detail below, along with DipGame, a testbed very similar to DAIDE but with it's own particularities. DipGame is the testbed used to support the development of the bot in focus in this dissertation.

2.2.1 DAIDE

DAIDE or Diplomacy AI Development Environment is a MAS testbed created by the DipAI organisation [DAI13, Dip13]. It provides the users with a server, several bots for testing purposes and some useful tools for the development of new bots. The whole DAIDE server and the bots are written in C/C++. DAIDE provides a very well specified and documented communication syntax used for exchanging messages between the players and the server, since even player-to-player communications are done through the server. DAIDE also categorizes the capabilities of communication of a bot according to Press levels, see Table 2.2. Each level contains a new type of message, each with a defined syntax.

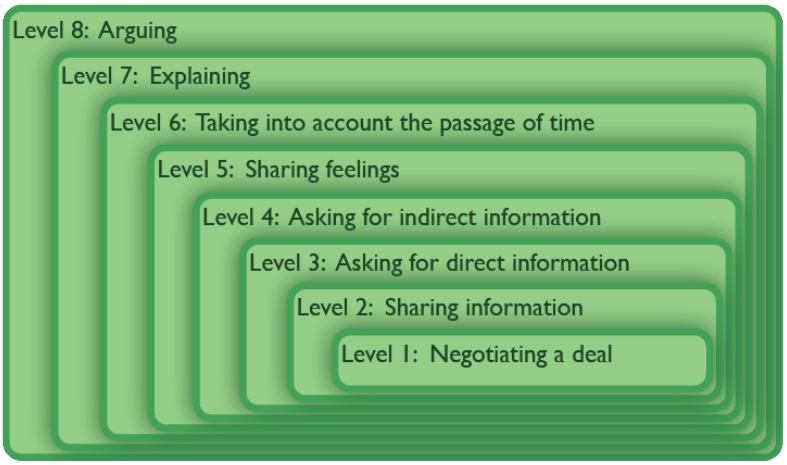

Figure 2.5: Layers of the L Language of DipGame

In order to build a solid bot, when implementing any layer, it is recommended the implementation of all previous layers. Currently most bots run in Level 0, which means the bots can only send game related messages to the server, in order to play the game. Some existing bots implement Level 10 and 20 with some difficulty. There are, however, some bots that implement solid strategies in Level 20 and even 30, described in Section 3.1.1 . There are no bots with a Press level higher than 30.

2.2.2 DipGame

DipGame is a MAS testbed, similar to and built on top of DAIDE, created by IIIA-CSIC of Barcelona [Fab13, IC13]. It is a project created to aid the scientific community in both testing and collecting results of Diplomacy games. DipGame has the advantage over DAIDE of being built in Java, therefore it is able to run in any OS. DipGame provides the users with a new set of tools, like a GUI interface to enable humans to play, a web server in which casual gamers can try competing against some provided bots, and an automated log system that tracks every interaction with the server, either orders or messages sent by players. DipGame has its own communication standard called L Language, see Figure 2.5 and for further details refer to [FS09, FS11]. Currently there are very few bots available for DipGame and even fewer with communication capabilities since the platform is still relatively new.

2.3 Summary

Diplomacy is a great environment for the creation of agents based on negotiation and there are multiple testbeds that allow developers to take advantage of it. These testbeds are in charge of ensuring and dealing with the complicated set of rules of the game, freeing the developers of that task and allowing them to focus on the creation of the bots. The next chapter presents related work

relevant for the approach followed in this dissertation, concerning existing bots and strategies used
2 in Diplomacy and other similar environments.

Chapter 3

Related Work

In this chapter it is presented a revision of literature of related topics and an overview of related work and state-of-the-art regarding implementations of existing bots and their negotiation strategies along with an analysis of solution search algorithms used Diplomacy and in similar scenarios. Finally a summary of trust assessing techniques is included.

3.1 Diplomacy Bots

This section comprehends the analysis of the most popular and pertinent bots developed for Diplomacy testbeds. These bots have different approaches to the presented problems and has been used for both inspiration and guidance during the course of this dissertation.

3.1.1 DAIDE Bots

The bots presented below were created for the DAIDE platform. Most of them are available in the DAIDE website for download along with many others for developers to test their bots against [DAI13]. Although DAIDE supports 15 levels for negotiation (see Table 2.2), most of the bots work with No Press, which is level 0. The higher Press level achieved, up until now, is Level 30.

DumbBot DumbBot is probably the most popular and common bot available. It was developed by David Norman as a challenge, in just two hours [Jon10, Nor13]. Although it was not optimized in any way and most of the values in the heuristics were chosen by chance, this bot 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 for orders.

The bot has been the target of many studies and has been used as a benchmark for testing other bots. It has very simple strategies to determine what actions to perform and obeys the following principle: *It is better to steal supply centres from strong enemies than from weak enemies.* This tactic is actually discussed in Diplomacy related literature and has been proved quite effective, since when a player controls a large amount of supply centres it can easily conquer even more. It is used for both attacking and defending regions: when DumbBot has to attack an occupied region,

it chooses the one owned by the stronger player; when it has to defend a region, it defends the one that will probably be attacked by the stronger player, with the purpose of keeping him from getting any stronger.

To determine the actions to make, DumbBot first assigns a value to each region of the map. This value depends on the player that owns the region, if owned, the number of its units and the number of enemy units, in the neighbouring regions – these units may be used for support in case of an attack. After all regions have an assigned value, each unit is given an action. This action is determined by a probabilistic selection from an array with all available actions sorted by ranking – better actions have a higher probability of being selected. If the occupied region has a higher value than all the neighbours, the unit is ordered to Hold. One of the DumbBot's bigger advantages is the simplicity of its strategy. Although it produces good results it still has a large room for improvement.

Albert Albert was developed by Jason van Hal and is, up until now, the best bot for DAIDE by far. It was inspired on the authors previous work, KissMyBot. It is the only Press Level 30 bot available and because of its efficiency and high performance, it has been used as a benchmark by many researchers who try to out-perform it [vH13].

BlabBot BlabBot is a Press level 20 bot created by John Newbury. It uses the heuristics of DumbBot and implements negotiation on top, to create a simple but effective bot. It uses the PeaceToAll strategy to send peace offers to all players (see Section 3.2). If any opponent accepts the peace offer, BlabBot decreases the value of the regions owned by that player. This way it is less likely to attack it and break the alliance. If no player accepts the peace offer, then BlabBot performs just like DumbBot [WCW08].

DarkBlade DarkBlade was built by João Ribeiro from University of Aveiro. It is a no press bot created with the objective of joining the best tactics and strategies used by other Diplomacy agents. It was developed with a architecture similar to that of the Israeli Diplomat ,(see Section 3.1.3), with the purpose of being easily extended and improved. Furthermore, the interactions between the different modules of the bot are done through a MAS of ,what the author calls, sub-agents. These sub-agents are the several parts that compose the bot and negotiate amongst each other to achieve a combination of actions to perform in each phase of the game. The bot successfully surpasses DumbBot and HaAI, achieving an increase of win of 53% over DumbBot, that was used as a benchmark [Rib08].

HaAI HaAI is a bot built by Fredrik Håård and Stefan J. Johansson [JHr05]. It has a distinct approach, because 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. Units can propose agreements with each other

to trade regions or suggest actions to others. Support moves are done by means of a contractual network. HaAI has two variations called *Berserk* and *Vanilla* which differ from each other in some coefficients and levels of aggression and caution. In tests, *Berserk* achieved a winning percentage of 18.4% while *Vanilla* scored 13.7% in games against DumbBots. HaAI managed to stay on the top list of bots until the year of 2006 [Rib08].

Dave de Jonge's Bot Dave de Jonge has done an optimization of 5 different variables regarding weights in the heuristics of genetic algorithms using different search algorithms. This attempt was successful and achieved an optimal version of the DumbBot without implementing new functionalities. This approach can be extended to embrace other variables [Jon10].

Deyllot's Database Bot Rui Deyllot from University of Aveiro suggested an approach for the decision making based on a move database. The major goal of the database is to provide the best set of actions for a given map and units with the goal of acquiring certain regions. Since it was impossible to create all possible game states, some level of abstraction was required using game state templates. The creation of the database was done after several games by the author himself [Dey10].

3.1.2 DipGame Bots

The existing bots for the DipGame platform are described here. Because the platform is very recent, there are not many bots available at the moment.

DumbBot DumbBot is an implementation of DAIDE's DumbBot. Although meant to be an exact copy of the original, this version has some minor changes like the removal of a type of action called Convoy, since DipGame does not support them. DumbBot does not negotiate.

SillyNegoBot SillyNegoBot was developed by Sylwia Polberg, Marcin Paprzycki and Maria Ganzha and is an extension of the SillyBot, a bot similar to DumbBot without communication capabilities. It adds Level 1 communication according to the DipGame L Language, it is built with a BDI architecture and has an internal division of responsibilities inspired in the Israeli Diplomat (see Section 3.1.3). The bot has proven to be successful when matched with the DumbBot but too naive when confronted with betrays, also, due to technical issues it is very unstable and often terminates unexpectedly. It uses the concept of personality with ratios for aggression/caution [PPG11].

3.1.3 Other Bots

Over the past years and with the immersion of Diplomacy as a research subject, many bots were developed along with a proprietary testbed. The bots presented below do not work with either DAIDE or DipGame but have interesting characteristics that deserve attention.

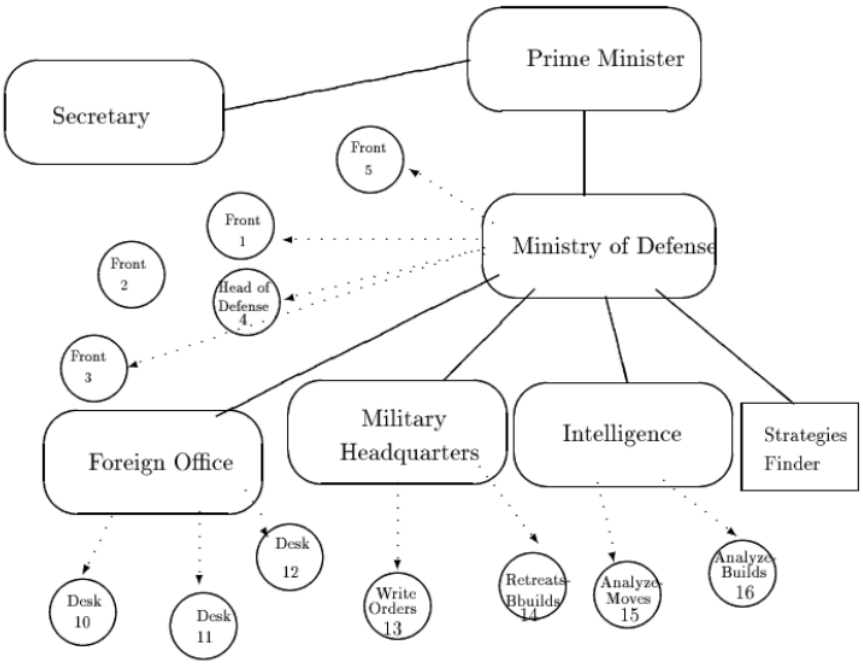

Figure 3.1: The architecture of the Israeli Diplomat, from [Rib08]

Israeli Diplomat The Israeli Diplomat was developed in 1988 by Kraus and Lehmann [KGL95, Sar87]. It uses an architecture that distributes responsibilities according to the nature of the tasks. It is divided into several nodes like the Prime Minister and the Ministry of Defence (see Figure 3.1). Its architecture has served as an inspiration for several other bots to come. The bot has several well designed strategies to deal with solution search and negotiation with opponents.

For every different negotiation in course, the Israeli Diplomat creates a sub-agent that will be responsible for that negotiation and is completely independent of all other sub-agents sometimes leading to simultaneous deals that contradict each other. It also has a sense of *personality* that allows the bot to be more bold or cautious depending of the situations and the course of the game. This also makes the bot's actions non deterministic.

The Bordeaux Diplomat The Bordeaux Diplomat was created by Loeb [HL95] and has a partitioned structure like the Israeli Diplomat that separates the negotiation from the solution search. The solution search ignores the World Power that owns each region and does an impartial evaluation of the action by the use of a best first algorithm called Refined Evolutionary Search.

The algorithm starts with a set of actions and mutates them until the best set of actions is found. It preserves the Nash equilibrium if it is found. For disbanding or building of units, the complete solution tree is searched since the computational effort is relatively low. The bot keeps a social relations matrix to determine the opponents that are more likely to betray him and break the alliance.

3.2 Negotiation Strategies

Negotiation plays an important role in Diplomacy, both in human playing and in bot development, given the size of the search space. In this section we focus on the main negotiation tactics that have been proposed. 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 the others. The presented tactics were already implemented and tested in Diplomacy bots.

3.2.1 Peace to All

This strategy is used by the BlabBot and consists on sending every player a Peace or Alliance request in an early phase of the game [WCW08]. This secures the player a set of alliances very soon. This blob of allied powers has a high chance of eliminating the players outside the group and once that is done, the bot will progressively betray the player that is considered the most convenient to leave the allied group. It usually targets the stronger player available.

3.2.2 BackStab

BlackStab is a tactic used by BlabBot for determining when to betray alliances and when these will be betrayed by the counterparts [WCW08]. It keeps a *threat* matrix between the player and the opponents and vice-versa, from the opponent to the player. This matrix represents the level of menace an opponent represents: the higher the value, the more likely the player is to betray the alliance.

If the value corresponding to the output threat, from the player to the opponent, is higher than the value from the input threat, from the opponent to the player, then the bot is more likely to betray than to be betrayed. The goal is to keep the output higher than the input. If the input is higher, the player is likely to be betrayed without being prepared, assuming the estimated values are correct.

During the game, the player should try and manage these values as much as possible, in an early and mid game phases. In an end game phase, when the player is forced to start betraying alliances, it should pick those that have a higher input value.

3.2.3 Power Cluster

Power Cluster is a simple approach to determine what World Powers to ask for alliance and which ones to keep the longest. It was built using clustering algorithms over several games and it rates the success of these alliances and how they contributed to the winning of the game. This technique tries to create small groups of powers that have a high probability of succeeding, if allied.

An example of the provided information would be: if the player represents France, it should try to ally itself with England, Germany, Italy, Austria, Turkey and Russia. The order presented

is meant to be an example of what the algorithm shows when used and is a suggestion of the best order possible for the creation of alliances.

3.3 Evaluation Heuristics

Evaluating board positions is crucial for effective Diplomacy playing. This section serves the purpose of providing an overview of the heuristics used by some bots already referred.

3.3.1 Province Destination Value

The Province Destination Value is used by DumbBot to assign a value to each region [Jon10]. It takes into account the player that owns the region, and the amount of allied and enemy units in surrounding regions. The value is determined by Equation 3.1, where r is the region, and the values pw, sw and cw are the weights associated with each parameter. The initial sum, from 0 to n, is used to search the neighbour regions within distance i.

$$DV_r = \sum_{i=0}^{n} pw^i.pm_r^i + sw.sv_r - cw.cv_r \tag{3.1}$$

3.3.2 Blurred Destination Value

This is a variation of the Province Destination Value that spreads the value of a certain node for its neighbours [Jon10]. This way, the surrounding regions reflect that either the region itself is valuable or is near a valuable region. The value assigned to the near regions can be obtained by several ways, either by applying a Gaussian Blur, a linear blur or by following any other formula. Figure 3.2 show an example of an application of a blur following the formula $\frac{x}{2^i}$, where i is the distance from the source.

3.3.3 Learning

Techniques based on learning are try to evaluate the value of each region according to experiences from previous games. The assigned value depends on the power the player represents and the impact of owning certain regions during the game. This enables a more realistic rating of the regions since it reflects the actual impact in practice instead of an estimated value. It reflects strategic points, choke points, regions with too many or too few neighbours.

3.4 Solution Search

In most areas of application of artificial intelligence, such as games or optimization, techniques of solution search are typically used as a way of reaching a feasible or optimal solution. Usually, solution search algorithms build and traverse a tree containing states of the solution space, however, in games like Diplomacy or Go, it is impossible to create the entire tree, due to the vast size

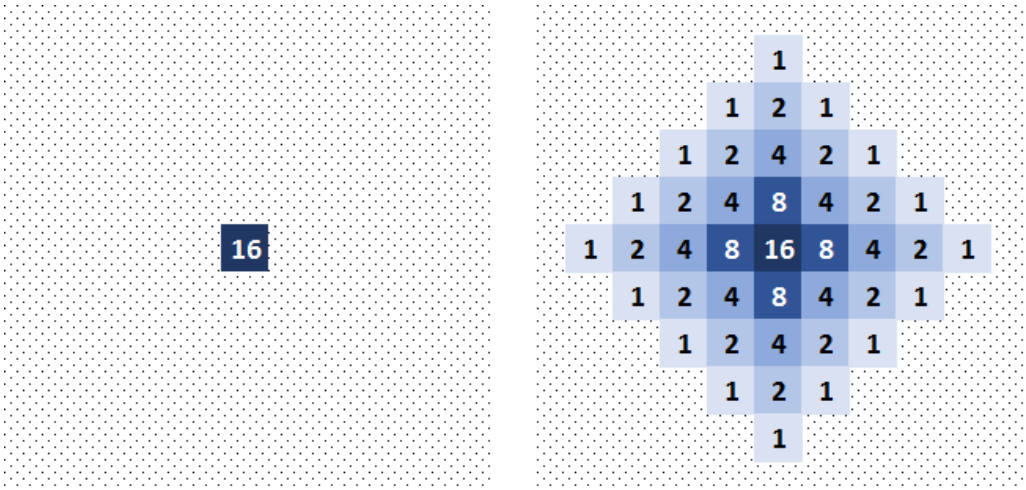

Figure 3.2: Example of Blurred Destination Value, before and after the application of blur, respectively.

of the solution space. This section contains an overview of solution search techniques used for similar problems where it is not possible to search the complete tree. These algorithms were not necessarily tested or meant for Diplomacy.

3.4.1 NB^3

NB^3 stands for Negotiation Based Branch & Bound, and relies on the standard Branch & Bound algorithm and introduces the concept of negotiation during the search process. It was developed by Dade de Jonge and Carles Sierra from the IIIA-CSIC of Barcelona [dJS11, JS12].

It was designed for a generic application and with no specific target in sight, but for testing purposes, a variation of the Travelling Salesman Problem [App07] was created and named Negotiation Travelling Salesman Problem [dJS11]. The new problem presents the same concept as the Travelling Salesman but in this case, with several salesmen, each trying to minimize their own trajectory cost.

In the problem, each node or city has to be visited by at least one salesman. Each salesman has one starting node where it begins and ends the journey and a set of randomly selected nodes. The agent must then negotiate with the other agents in order to trade nodes with each others. They can exchange every node except the starting node.

The authors propose the usage of NB^3 in Diplomacy since in both scenarios the effects of the actions done by one agent depend on the actions of other agents. Also, both scenarios have a negotiation phase and a posterior execution phase, in which all players act simultaneously. Although agents in the Negotiation Travelling Salesman Problem are assumed to keep their agreements, the scenarios are still quite similar.

NB^3 uses progressive negotiation as it scans the solution tree, in order to prune certain branches

it considers will never be visited. This negotiation is initiated by both the agent and its competitors. Each time a negotiation ends, the new agreement is used as a restriction that must be considered, and the tree is pruned of all the branches that produce infeasible solutions.

The negotiations might include discussion and argumentation according to the scenario, however, the algorithm only uses the final agreement. For every accepted agreement, the environment is changed, assuming the action was already made. This way, the agent can search its own actions in a more likely game state.

The need of negotiating during the search, and not only after the optimal solution has been found, comes from the fact that if the agents wait too long to propose their terms, the other agents might already have signed contradictory agreements with other agents, leaving no feasible solution but the initial one. There is a trade-off between reaching the optimal solution and being available for commitment. Solving this trade-off is the key aspect for successfully applying this algorithm.

To determine which node to split, the algorithm uses a best first search supported by an heuristic that represents the value or gain for the agent itself. When a node that is considered optimal so far is reached, and if the node depends on other agents actions, a negotiation is proposed between the concerning agents with the objective of securing that node. Because of the trade-off, it is often the case that the algorithm reaches a node better than a previous one, but which is not able to secure because of commitments already made.

During the process of searching the tree, values for upper and lower bounds and intermediate values are kept showing the worst, best and current case scenarios possible. The global upper bound is the value the agent could achieve with no cooperation. The intermediate value represents an estimation of the value obtained assuming all agreements made are kept and no other agreements are made. The lower bound represents the current best possible value with the agreements made. The algorithm is presented for a minimization problem. To adapt it to a maximization problem, the values of upper and lower bound are switched.

3.4.2 Minimax

Minimax, also called Min-Max, is a decision rule used in solution tree search that tries to minimize the loss of a certain state while maximizing the profit [RNC+95]. It was first created for two players zero-sum games. These are games in which if a value V is assigned to each player according to its outcome, the other player will have value -V, thus, the sum of all players' values will be zero. This means that every time a player gains some advantage, the other player loses advantage in direct proportion.

During the search, the tree is created having alternate layers of min and max levels that represent the opponent and the player itself, respectively. In every min layer the objective is to find the lower value, that represents the opponent choosing its own best action: the opponent's best action will be the worst action for the present player. In every max layer the goal is to find the higher value that represents the player's own best rated action. Therefore, the min layer can be seen as a max layer with inverted values – this variation is called the NegaMax. The final value can be computed by Algorithm 1.

Algorithm 1 Negamax pseudo algorithm

```
negamax(node, depth) :
if depth = 0 then
    return heuristic(node)
else
    α ← −∞
    for all child : node.children do
        α ← max(α, −negamax(child, depth − 1))
    end for
    return α
end if
```

This algorithm can be further improved by implementing *alpha-beta* pruning, which consists in keeping track of two values that act like upper and lower bounds. By updating these values and comparing them with each other it is possible to see that there is no need to search the remaining tree since none of the following nodes will be visited. It can be modified to work with N player games like Diplomacy, leading to a *max-min-min-min-min-min-min* tree, for the example of a game with seven players. Since Diplomacy's actions are simultaneous, the state of the game state will not update in every layer, but in every seven layers that represent a turn.

Minimax is successfully used in many games and other areas of application of artificial intelligence, however, since it relies on search in depth by default, it is not indicated for games like Diplomacy, due to its large branching factor, therefore the amount of game states, even in very low depths, will become impractical.

3.5 Trust

Diplomacy is a MAS where agents are selfish and have individual goals rather than common ones. However, agents benefit from cooperation in a way that it helps them achieve their own goals, with less cost than individually. This cooperation is often based on contracts between agents. Contracts are commitments regarding an action or set of actions to be fulfilled by the involving agents. Although contracts should benefit both participants, agents can opt to act differently than what was stipulated, by having another contract that is more beneficial, for example. Therefore, the need for trust arises.

Trust can be seen as the predictability of an action or an outcome. Trust does not mean *to trust*, it means to believe something will happen. For example, in a supply-demand chain, if a buyer knows the supplier always delivers one day late, he can *trust* that the supplier will be late. Trust can be used to manipulate the outcome to our favour: in the example given, the buyer can schedule the delivery one day before the actual deadline.

3.5.1 Dynamic Environments

Trust often takes time to build because there is the need of several interactions with a certain amount of associated risk. In highly dynamic environments, where agents can enter and leave the system at will, trust can be hard to evaluate accurately. To tackle this problem, C. Burnett proposed a stereotype model, which allows agents to build trustworthy relations based on visible aspects [Bur11] . This stereotype approach allows the agent to create a set of templates characterized by certain parameters. During the execution, the agent tries to match its opponents with the stereotypes and address them according to the trust held in the stereotype. This approach allows a quick estimation of trust without the need of prolonged assessment of the opponents and the environment.

During the initial contact with other agents, when there is no trust or evidences to match the stereotypes, the author proposes the use of control to improve the speed of trust building. Control can be described as the use of extra methods that ensure and evaluate the fulfilment of the agreements and the adjustment of expectations to deal with possible deviations. The usage of control means an increased overload of the system and the agent, however, in early states of interaction there is an urgent need to assess the surrounding agents. As trust begins to build, the usage of control can decrease until the point where there are enough means to evaluate trust by the use of standard methods.

3.5.2 Trust Assessment over Time

In most trust assessment techniques, an assumption is made that the contracted actions are meant to be executed immediately. Nevertheless, in some scenarios this is not true and the agreements are only fulfilled after some time; therefore, the agents should take time into account when assessing the completion of the agreed terms. If the contract has a deadline this evaluation is easier to make, however, deadlines are not always part of the contracts and the agents tend to assume the contract was not fulfilled and the involved agents are not trustworthy.

The concept of trust assessment over time has been the subject of study for some time. C. Sierra proposed a way to evaluate the fulfilment of an agreement over time and its following consequences [SD13]. This method takes into account the fulfilment of an agreement and its following support, for example, a contract can specify the purchase of an item in a given time, and although the item was delivered within the time limit, the quality of the item was not the expected.

3.6 Summary

Dave the Jonge, who is involved in the DipGame platform, had been developing a DipGame bot during the course of this dissertation. This bot could be used for testing since there is a lack of test subjects, besides DumbBot. Unfortunately, this work was not finished by the end of the testing phase of DipBlue.

The majority of Diplomacy bots and strategies focus on solving the problem through techniques related with solution search. Although some approaches use negotiation, it is only used as a mean to help solution search, these approaches typically fail to take advantage of the full potential of negotiation, since they focus in collecting information and not in controlling the actions of the opponents. The next chapter describes the bot developed in this dissertation which focuses on communication and uses negotiation as its main tool to obtain advantage over its opponents.

Chapter 4

DipBlue

This chapter focuses on describing the proposed DipBlue, a bot for Diplomacy. The overall implementation of the bot, its architecture and tactics and heuristics will be presented.

DipBlue has has been named in honour to the super-computer DeepBlue which, in 1997, played and won a game against Gary Kasparov, the chess World Champion at the time [Cam01], becoming the first computer ever to win a match against a World Champion under tournament rules. The name also refers to the platform it is built to, DipGame.

DipBlue is characterized as being an artificial player for the Diplomacy game built with the purpose of assessing and exploring the impact of negotiation in a game that relies on communication by default. Since the main difficulty when creating a Diplomacy bot is the size of the search tree, a different path was taken in order to overcome the existing challenges. Accordingly, DipBlue uses negotiation as its main tool to gain advantage over its competitors and applies trust reasoning to understand and react when betrayed.

Regarding implementation, the bot uses the Player class, packed in the *dip* library developed by Angela Fabregues at IIIA-CSIC, which handles the interactions between the player and the game server and offers an interface with the negotiation server. This library provides both an updated version of the world reflecting the current state of the game and a report of all the moves in the end of each phase of the game. In addition to the information provided by the library, DipBlue stores extra information to better help determine its actions, the most important of which are the trust ratios. These are associated to all opponents and reflect the status of their relationship, which are key to all negotiation-based logic in DipBlue.

Next, the architecture of the bot, the negotiation tactics and Advisers used by DipBlue and, finally, the specification of the different archetypes will be described.

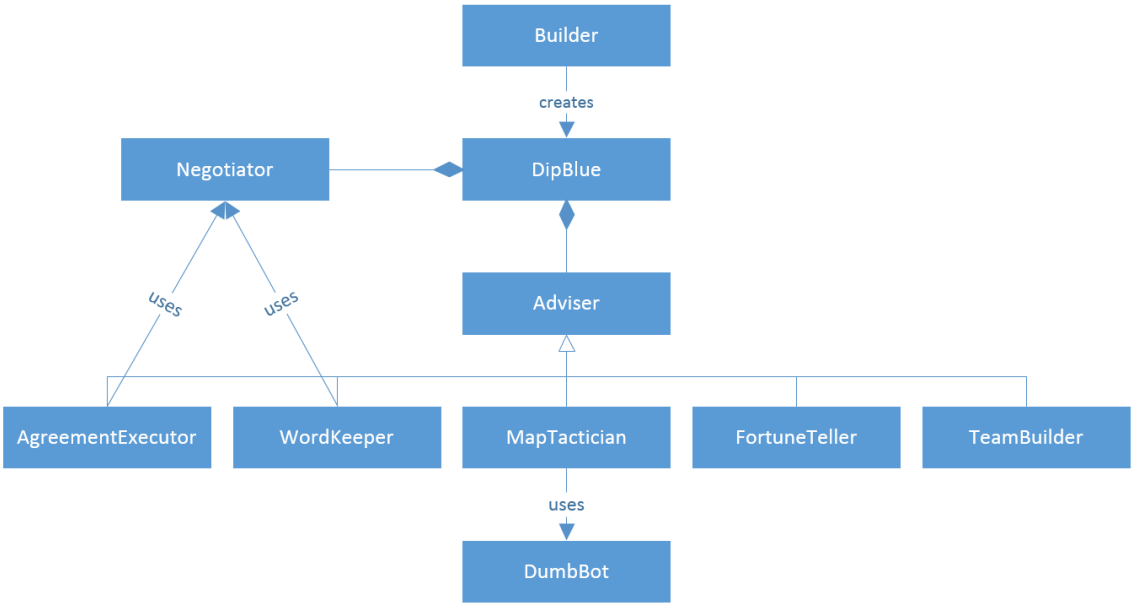

Figure 4.1: An overview of the DipBlue

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 system, which evaluates and determines the set of moves in each turn, from different perspectives. Figure 4.1 shows an overview of DipBlue's architecture, displaying the main parts of the bot and the interactions between them. This modular implementation allows an easy customization of the bot resulting in a vast array of possible configurations of bots, that differ in their capabilities and behaviours. Also, since the source code will be publicly available, it will offer the community an accessible interface that allows to change and create new modules for DipBlue.

The goal of the division into modules is to enable the creation of Advisers separately from the bot itself. With this implementation, the bot contains the interface with the DipGame platform and is responsible for receiving and sending orders and messages, while the Advisers determine the orders to be made.

Although the decision making part of the bot is modular and independent, the negotiation is done by a single node called DipBlueNegotiator. This node is responsible for handling received messages and for determining outgoing messages. All tactics regarding negotiation are implemented in this Negotiator but, since the deals made in Diplomacy have no real impact in the game, the result of the negotiation only affects the actions made by the players. Therefore, the agreements made have to be taken into account by some Advisers, otherwise they will be ignored.

Level 1: Negotiating a deal
$L_1 ::= \text{propose}(\alpha, \beta, deal_1) \mid \text{accept}(\alpha, \beta, deal_1) \mid$
$\text{reject}(\alpha, \beta, deal_1) \mid \text{withdraw}(\alpha, \beta)$
$deal_1 ::= \text{Commit}(\alpha, \beta, \varphi)^+ \mid \text{Agree}(\beta, \varphi)$
$\varphi ::= \underline{predicate} \mid \text{Do}(\underline{action}) \mid \varphi \wedge \varphi \mid \neg\varphi$
$\beta ::= \alpha^+$
$\alpha ::= \underline{agent}$

Figure 4.2: DipGame's Language Level 1. *Predicate* is either peace or alliance, *action* is any order performed by a unit and *agent* is a power.

4.2 Negotiation

Negotiation is the main aspect of DipBlue and is what gives it the advantage over players who rely on solution search. It has the ability to communicate using messages of the DipGame's Language Level 1 whose format is shown in Figure 4.2. All negotiation tactics are implemented inside specific routines that are called throughout each phase of the game. In some phases, negotiation is simply skipped, as in DipGame it is only possible for players to negotiate in Spring and Fall phases. Concerning incoming messages, they are handled by a listener, therefore, receiving and sending messages are two separate and distinct tasks.

4.2.1 Trust Ratio

One of the key components of the negotiation tactics is the *trust ratio* held by the bot. This ratio reflects the relationship between the player and each opponent. Initially, all players have a ratio equal to one, which means they are all neutral. This ratio is converted into a friction ratio by inverting it, so that $Friction = \frac{1}{Trust}$. This ratio is used by the bot to reflect the making of alliances and to increase the odds on the fulfilment of deals. It also determines when certain deals are accepted or rejected and whether they should be performed.

In the course of time and along with the interactions between players, these ratios rise and fall according to what happens between the player and its opponents, as the following examples demonstrate. The friction ratio corresponding to a player decreases if the given player does not attack DipBlue. This is considered a friendly attitude, hence increases the trust in the player. On the other hand, when a player attacks DipBlue or breaks a previously made deal, the trust diminishes. The amount of increase or decrease of this ratio is linked to the current trust held in the player: players currently considered untrustworthy have lower impact in this ratio, which means that both increases and decreases are made in a lower quantity; players considered trustworthy have a higher impact, which means that both positive and negative actions have a high impact. This is used to reflect betrayals during the game, since an attack made by an ally has a higher increase of friction than the same attack made by a current enemy. In addition to the previously described ways to change the ratio, the values are inflated when two countries are in either war or peace/alliance. In cases of peace, the trust is accounted as double of the real value and in cases of war the trust is considered half of the real value.

DipGame's Language Leve 1 is composed primarily by Peace, Alliance and Action requests, all of which are detailed below along with the corresponding tactics. Although these messages are supported by the DipGame platform, the game server does not perform any type of validation concerning the content or the future fulfilment of the deals. Such as in the original board game, the messages only have symbolic meaning between the parts who exchange it. For this reason, there is no strict meaning for any message, each player assigns the meaning it considers appropriate. I.e., there are no specifications on how a player should behave after accepting a peace request besides common sense and the general understanding of the word *peace*.

Along with the trust ratio, there is a *state* associated with each opponent that alse reflects the relationship with the opponents, this state is originally *neutral* and may change to *war* or *peace* according to the trust ratio and the negotiations. This state is used to enhance the impact of the ratio, by increasing its effect when assessing actions related to the given opponent.

4.2.2 Peace

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. When DipBlue sends or accepts a peace request, the state regarding the correspondent player changes to peace, inflating the effective trust ratio.

Peace messages are sent to all negotiating players in the beginning of the game in an attempt to reduce the probability of conflict with the most players possible. Nevertheless, conflicts will need to exist for the game to continue. Therefore, DipBlue opts to break truce with the player considered to be the least beneficial. The process of choosing who to break truce with comprehends the number of supply centres held by the other powers and the proximity between the power under analysis and DipBlue in the map. This strategy has proven to be useful in an environment where trust between players is not taken into account. However, if trust assessments are used amongst the opponents, breaking truce may have more severe repercussions by lowering DipBlue's perceived trust by other players and increasing the risk of loosing other alliances.

4.2.3 Alliances

DipGame implements the Alliance request using two clusters of powers called 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 and the number of its supply centres is reduced, the new strongest non-ally power is targeted and the cycle restarts.

DipBlue accepts requests from other players if they are in a state of peace and if the targeted enemy is not an ally itself. When a new alliance has started, all enemy players' states are changed to war, thus reducing the trust ratio and increasing aggressiveness towards them.

4.2.4 Order Requests

An order request is sent to a player containing an order regarding a unit of that player. It has the purpose of suggesting the other player orders for their 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 previously selected action for that unit. The value of the requested order is calculated in the same way as other orders and is scaled with the trust ratio of the sender, i.e., players with a higher trust value have a higher possibility of having their requests accepted.

A skilled player may use a Slave archetype as an extension of himself through the use of order requests, by making it execute any actions he considers beneficial and, eventually, requesting suicidal actions when only the two players remain playing. See Section 4.4 below for a better explanation of the Slave archetype.

4.3 Advisers

The suggested architecture for DipBlue resembles the Israeli Diplomat in such a way that every task and responsibility is assigned to a specific part of the bot. This modularity is meant to be used as a plugin system. Each part of the bot is called Adviser and, although they operate independently, some have the ability to operate together, and all of them can be inserted and removed from the bot willingly. In the process of determining the actions, the opinions of all the Advisers are taken into account in order to reach a final decision that tries to satisfy them all.

One of DipBlue's small and yet crucial aspect is the way actions are selected to be performed. The Adviser mechanism,allows the evaluation of all possible actions for all units. After all actions are assessed, one action must always be selected for each unit to perform, which cannot be simplified to picking the best rated action for each unit, as this selection may trigger standoffs. Standoffs occur because there is the possibility for more than one unit to be assigned to attack the same region, which would result in a conflict and neither unit would be successful. To prevent standoffs and achieve the best possible set of actions, all actions are sorted according to their value. Then, each one is added to a new set in order to be executed, if they fulfil two requirements: belong to a unit that does not have an assigned action and do not interfere with the actions previously selected. In the end of the selection, units that were assigned to hold are re-assigned to a support action to a nearby unit, if such action is possible.

The method used to calculate the value assigned to each action is a weighted accumulation similar to a voting system. The values each Adviser returns are accumulated in order to create the meaning of value for each action: see Equation 4.1, where n is the number of Advisers, w^i is the weight of each Adviser and v^i_{Order} is the value each Adviser assigns to the given action. Although

the default usage of the values returned by an Adviser is to add to the previous value from other Advisers, some multiply the returned value with the previous ones, resulting in a sort of scale of the value, therefore, the order in which the Advisers are executer is important. This is done when the value by itself has no meaning, for example, the probability of an action. This approach comprehends the opinion of each Adviser attending to their weights. Since each Adviser has its own purpose, all opinions are taken into account in the final group of orders.

$$V_{Order} = \sum_{i=0}^{n} w^i . v^i_{Order} \tag{4.1}$$

Another possible solution to determine the orders would be the creation of an internal MAS environment, in which each Adviser tries to impose its own opinion by arguing and trying to convince the other Advisers. The Advisers would negotiate with each other in order to achieve a solution that satisfied all of them. This implementation turned out to be too overwhelming and effortful and does not seem to be the optimal way of gathering a set of orders to execute since, in the end, the selected orders would belong to several different Advisers and could be incoherent.

Initially, all Advisers have equal weights but, further in the process, the weights will be adjusted in order to achieve an optimal result. Along with the weights DipBlue holds regarding each Adviser, the Advisers themselves have intrinsic parameters that can be adjusted for different behaviour variations. This adjustment of several coefficients allows the creation of behavioural archetypes and personality, such as Aggressive, Naive, Friendly and Vengeful. These personalities can be adjusted manually in a preliminary phase and then optimized by a method of solution search, as evidenced in the work done by Dave de Jonge with genetic algorithms [Jon10].

Next, a short description of the created Advisers is provided.

4.3.1 MapTactician

The MapTactician is the base Adviser that serves as a starting point for all the following Advisers to work upon. Since the main goal of this dissertation regards negotiation and trust and not simply the creation of a bot capable of playing the game, this Adviser has the purpose of being the skeleton for DipBlue, placing it as close as possible to the most common Diplomacy bots. For this, the DumbBot was chosen since it is the only bot available for DipGame. Using the extended documentation of the DumbBot's heuristics and inner workings done by Dave de Jonge [Jon10], it was possible to create a similar version of the DumbBot inside this Adviser. However, after some testing it turned out that the performance of this implementation was significantly below expected and considerably below the performance of DumbBot.

As a mean to bring the rate of success closer to the expected, the heuristics of the DumbBot were used directly, thus making the evaluation given by this Adviser the same as DumbBot's. DumbBot uses the Province Destination Value, described in Section 3.3.1, which assigns a value to each region according to several aspects of the region, its neighbours and its owner. This value is returned by the MapTactician.

Although the evaluation method is the same, there are more aspects in DumbBot besides the evaluation of the regions. Even when the heuristics are mimicked, the behaviour of DumbBot and the DipBlue using MapTactician are not the same.

4.3.2 FortuneTeller

FortuneTeller aims to give a probabilistic view of the evaluated Move actions. Originally, this was done by means of an Adjudicator made and provided by Dave de Jonge. Since Diplomacy has a very complicated set of rules with many exceptions and precedences between them, the task of determining if one action in a given set of actions is going to be successful or not, is not trivial. As seen previously, the size of the game tree is astonishing, thus only a small part of the tree could be searched. However, even by narrowing the search space to neighbour units of the ones controlled by DipBlue, the time spent on calculating the outcomes for the different combinations made the use of this technique impractical.

As an alternative to the Adjudicator, a simpler and more error prone version was created, disregarding the possibility of chain actions that may nullify each other. This new version executes much faster than the previous one – reducing the average time of execution from near 15 minutes to less than 2 minutes in a full regular game – although it returns more optimistic and thus less realistic probabilities of success.

4.3.3 TeamBuilder

TeamBuilder's role is to promote Support actions. This is accomplished by increasing the value of both the Support and Move action that is in need of support, because of the nature of its action. Some units may forfeit their original actions to support some neighbours with a high need for support and a value higher than its own original action, further in the process of choosing the actions for each unit. Changing the weight of this Adviser results in a higher cooperation in attacking moves, resulting in a state where almost every Move action has one or more supporting units.

4.3.4 AgreementExecutor

The AgreementExecutor ensures that the agreements made with other powers are fulfilled. By calculating the value of the agreed actions, this Adviser is able to assess the value of a deal and increase the value of the actions that allow the fulfilment of the deal. The value assigned to each deal is calculated as show in Algorithm 2. It is directly proportional to the trust ratio with the power with whom the deal was made. As this ratio increases when the involved powers are in a peace or alliance state and decreases drastically when the powers are in war, a deal can be proposed and accepted when the powers are in a friendly state but then be poorly rated because of the decrease of the trust between both parties. Because of this possible oscillation of the ratios, certain deals may be broken shortly after they were made without an explicit and premeditated intention of betrayal.

By adjusting the weight of this Adviser, it is possible to change how DipBlue behaves regarding the fulfilment of deals.

Algorithm 2 Agreement Executor's evaluation of an action

```
agreementExecutor (action) :
if isInPeace(action.owner) then
    return peaceScalexaction.owner.ratioxevaluate(action)
else
    if isInWar(action.owner) then
        return warScalexaction.owner.ratioxevaluate(action)
    else
        return action.owner.ratioxevaluate(action)
    end if
end if
```

4.3.5 WordKeeper

WordKeeper is the Adviser in charge of reflecting the influence of trust and friction regarding each opponent. For each offensive order WordKeeper returns the friction ratio regarding the opponent (see Section 4.2.1). As this ratio increases when the bot is attacked or betrayed or when is in war with someone, and decreases when is not attacked or when it is in peace, this Adviser simply returns the ratio multiplied by the value from the previous Advisers. This way the player has a higher probability of attacking regions held by powers who have a low level of trust or to move towards them.

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. The configuration obeys the to following structure:

Communication The ability to receive messages.

Active Communication The ability and strategy to send messages.

Strategy If *Balanced*, the bot judges if the proposed deals are beneficial, this assessment is based on the proposed terms and on the world power that makes the proposition.

If *AlwaysYes*, the bot always agrees with any proposal.

Trust The ability to perform trust reasoning regarding deals and the opponents actions in the previous rounds.

Advisers The list of Advisers used by the bot

The archetypes and their characteristics are described below.

4.4.1 NoPress

NoPress is the simplest bot and the most similar to DumbBot. It only uses Advisers related to the map and its units, and lacks the ability to handle communication, negotiate or perform trust reasoning based on either deals or actions. It is expected that NoPress has a performance identical to DumbBot. Table 4.1 shows the configuration of the bot.

Communication	No
Active Communication	n/a
Strategy	n/a
Trust	No
Advisers	MapTactician FortuneTeller TeamBuilder

Table 4.1: Configuration of NoPress

4.4.2 Slave

Slave has more capabilities than NoPress as it has the ability to receive messages sent from other players. Although technically it also has the ability to send messages, it is not used with the purpose of restricting the behaviour of the bot. Slave makes the same strategic evaluation of the actions as NoPress. In addition, it receives requests of peace, alliances and actions and follows them blindly, as long as they are valid and executable, without making any assessment of the value of the request. All agreements have higher priority as compared to the actions determined by the bot itself, hence all deals will be fulfilled. Ideally, this is the best bot to have as an ally since it complies everything it is told; however, the performance depends on the expertise of who makes the requests/orders. Table 4.4 shows the configuration of the bot.

4.4.3 Naive

Naive is similar to Slave with the exception of being endowed with the ability to propose deals to other players. Also, it weights incoming requests in order to determine their advantages. The likelihood of a request being accepted depends on the content of the request itself and the player who makes the request. the two aspects are taken into account to determine the value of the deal.

Communication	Yes
Active Communication	No
Strategy	AlwaysYes
Trust	No
Advisers	MapTactician FortuneTeller TeamBuilder AgreementExecutor WordKeeper

Table 4.2: Configuration of Slave

The value of deals proposed by allies or players with very high trust ratio are inflated and the opposite also occurs; requests made by players the bot is in war with, are almost always rejected. Although this bot is very balanced and capable of social interactions it still lacks the ability to perceive when agreements are not fulfilled, which makes the bot highly innocent or, as the name indicates, naive. Table 4.3 shows the configuration of the bot.

Communication	Yes
Active Communication	Yes
Strategy	Balanced
Trust	No
Advisers	MapTactician FortuneTeller TeamBuilder AgreementExecutor WordKeeper

Table 4.3: Configuration of Naive

4.4.4 DipBlue

DipBlue is the main and most complete bot, combining the abilities from all bots mentioned above plus the ability to perform trust reasoning, using the model described in Section 4.2.1. Due to the

use of trust, this bot is expected to have the best performance and to be more capable of dealing with allies that may betray it. Table 4.4 shows the configuration of the bot.

Communication	Yes
Active Communication	Yes
Strategy	Balanced
Trust	Yes
Advisers	MapTactician FortuneTeller TeamBuilder AgreementExecutor WordKeeper

Table 4.4: Configuration of DipBlue

4.5 Summary

DipBlue has the purpose of using negotiation with its opponents as a means to achieve superiority. Its modular architecture allows the creation of different bots and simplifies improvements and the creating of new modules. The DipBlue approach has been tested in several scenarios presented in Chapter 5. The obtained results were confronted with the hypotheses drawn in Section 1.2, the main conclusions being presented in Chapter 6.

Chapter 5

Experiments and Results

This chapter focuses on the evaluation of the bot and the different test scenarios and their results. It explains the different tests made and the expected and obtained results in each of them. Finally, the proposed hypotheses are reviewed and assessed accordingly to the obtained results.

5.1 Set up

In addition to DipBlue, and to help improve the process of the design of experiments and gathering of results, some tools were created, such as a Scout to monitor the process of the games, and several changes were made to the game launcher and the Player classes, both provided with DipGame and its libraries. The major changes and why they were made are explained in the sections below.

5.1.1 Game Launcher

The original game launcher, provided with DipGame, relies on a graphical user interface, or GUI, to select the players, launch the game and track its progress. This launcher is packed inside a jar file and after being launched requires a manual selection of all the opponents and the posterior launch of the DipBlue bot from a command line or from inside an IDE. Another issue with the original launcher is when one or more players block or abort unexpectedly – the launcher and the other players continue running in background and must be stopped manually since the launcher ceases to respond. Although very useful for occasional testing or for specific analysis using the graphical map, this launcher proved to be a blockage when the need to run several games arose.

To overcome this obstacle a new launcher was created that is able to launch games from Java code and to determine which bots will play. This eliminates the need to use separate tools and to switch environments when testing. Also, since all processes are launched from code, it is possible to track and kill them if there is the need for it. To launch one or more games one simply needs to specify the bots that need to be tracked or debugged and the launcher automatically launches the remaining positions with DumbBots. If several games are to be launched each game starts in succession and produces an individual log.

Overall, the changes made to the launcher allowed a faster and more flexible development and debugging of the bot, specially since most tests were made in batches of seventy games in a row, which would be absolutely impractical with the original launcher. The reason why each configuration was tested with seventy games is because there are seven world powers, and seventy is a multiple number of seven, allowing all players to play the same amount of games with each world power, as well as all opponents; and because making multiple executions allows the collection of several results and the evaluation of the average performance of the bots.

5.1.2 Player

The Player is the class extended by DipBlue that performs all the basic interactions with the game server. Originally it provides a rudimentary dump of all messages sent and received to the game server and the negotiation server. This log contains all the information about the game but in a very raw state that hampers its usage.

In order to provide better logs some changes were made to the Player directly to collect information about the moves that are made, the rate of success of the attacks done in the previous round, the year in which a player loses the game or the amount of supply centres in case of a draw.

5.1.3 Scout

To further improve the logging of the games an extra bot called Scout was developed that extends the Observer instead of the Player. This way it is allowed to enter games without counting as a player. This bot has the single purpose of logging and monitoring the game, being very helpful to better understand how some opponents behave since they don't necessarily create logs of their own.

5.2 Scenarios

In order to test DipBlue and to validate the proposed hypotheses, several tests were made using full games in order to assess the performance and the efficiency of the decision making of the bot. Given the complexity of some tactics and behaviours, it would be beneficial to test isolated scenarios to assess a single move made or message sent, however due to the restrictions imposed by the DipGame platform, these types of tests were not possible to perform and the effort of further changing the game server proved to be disproportional to the gain it would return.

Since the goal of the dissertation is to build a bot that takes advantage of negotiation and trust, some scenarios were designed to include environments where the bot can negotiate with other players and even when it cannot, to understand how it performs in different situations. To do this, the tests cover situations from having no one else to negotiate with to having all the opponents with negotiation capabilities.

For every different scenario and configuration, seventy games were made and their results were stored in a separate file. The values are output by the bots with data concerning the game and their own performance. The variables whose values are recorded are described in Table 5.1.

Variable	Description
GameNumber	Number of the game
BotName	Name of the bot
Power	World power the bot was assigned to in the beginning of the game
Year	Year in which the game ended
Position	Position in which the bot end the game in
Winner	Which power won the game
YearsByPower	Year in which each power lose the game
PositionByPower	Position in which each power end the game in
AllyPowers	Power with whom the bot was allied with
AllyDistance	Average distance of the allies to the bot
EnemyPowers	Power with whom the bot was in war with
EnemyDistance	Average distance of the enemies to the bot
#CommPlayers	Number of players able to communicate
AcceptedMessages	Number of sent messages that were accepted
RejectedMessages	Number of sent messages that were rejected
#Holds	Number of hold actions performed by the bot
#Moves	Number of move actions performed by the bot
#MovesItself	Number of move actions performed by the bot, attacking itself
#MovesAlly	Number of move actions performed by the bot, attacking an ally
#MovesEnemy	Number of move actions performed by the bot, attacking an enemy
#MovesNeutral	Number of move actions performed by the bot, attacking a neutral power
#MovesEmpty	Number of move actions performed by the bot, moving to an empty region
#SupportItself	Number of hold actions performed by the bot, given to its own units
#SupportAlly	Number of hold actions performed by the bot, given to allied units

Table 5.1: Variables collected by each bot regarding itself, its opponents and the game

The scenarios used to test the bots are presented in Table 5.2. For each scenario a description of the players involved is given, along with its purpose.

Scenario	Bots	Purpose
Scenario 1	1x NoPress 6x DumbBot	Test the performance of a single bot without negotiation or trust
Scenario 2	1x DipBlue 6x DumbBot	Test the impact of a single bot with trust but without negotiation
Scenario 3	1x Slave 1x Naive 5x DumbBot	Test the performance of a team containing a bot without free will, without the possibility of betrayals
Scenario 4	1x Slave 1x DipBlue 5x DumbBot	Test the performance of a team containing a bot without free will, with the possibility of betrayals
Scenario 5	1x Naive 1x DipBlue 5x DumbBot	Test the performance of a team containing a bot without trust reasoning, with the possibility of betrayals
Scenario 6	2x DipBlue 5x DumbBot	Test the performance of a team with two bots with trust reasoning and the possibility for mutual betrayal
Scenario 7	7x NoPress	Test the behaviour of multiple equal bots without negotiation
Scenario 8	7x DipBlue	Test the behaviour of multiple equal bots with negotiation, trust reasoning and the possibility for betrayals
Scenario 9	2x DipBlue 5x NoPress	Test the performance of a team with two bots with trust reasoning and the possibility for mutual betrayal when all opponents are known

Table 5.2: Description of test scenarios

The expected and actual results of the various test scenarios will be presented in the next section along with a thorough analysis that tries to find possible explanations of the results.

5.3 Results

After collecting results from several games done in all previously described test scenarios, the data was treated and analysed in order to extract useful information.

5.3.1 Overall Performance

The most relevant result is the position in which the bot ends the game, since it provides the most direct insight of the bot's performance. In games made with seven DumbBots, the average position

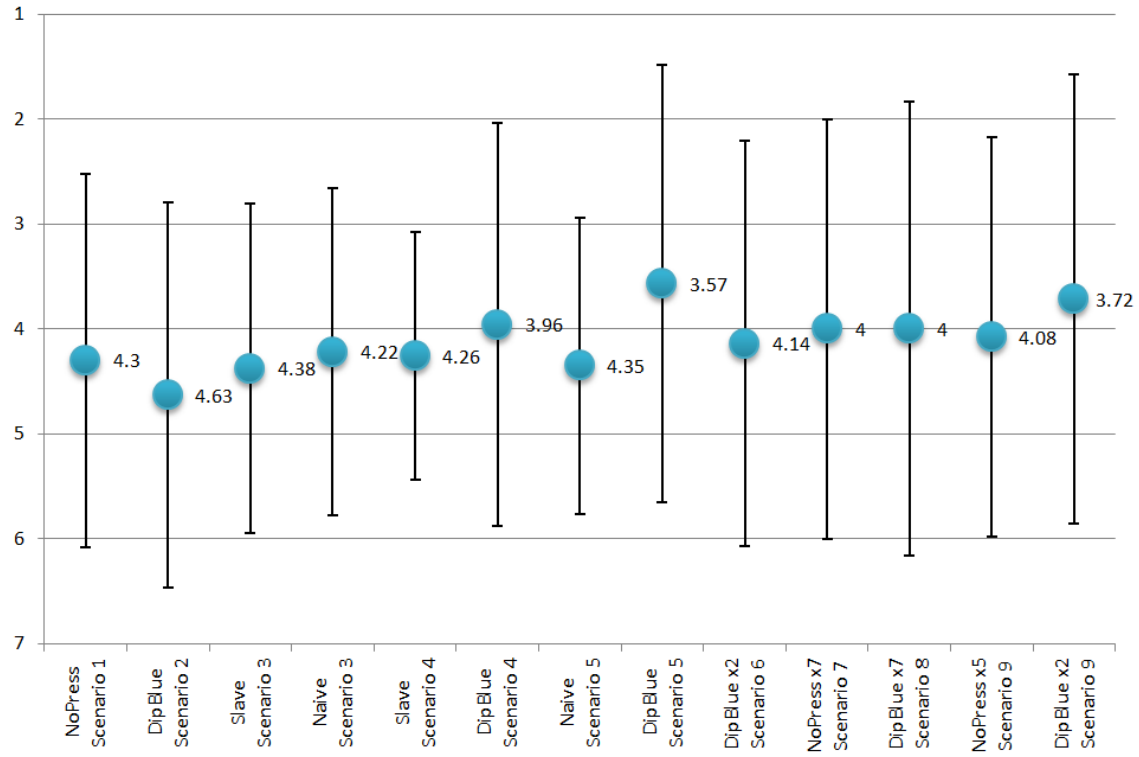

Figure 5.1: Average and standard deviation of the final position of the bot in each scenario.

is 4th place, since all players have equal performance, therefore, there is an even distribution of wins.

By analysing the average position obtained by the NoPress in scenario 1, which is 4.3, as shown in Figure 5.1, it is possible to conclude that the performance of the bot is lower than the performance of DumbBot. Although NoPress uses the heuristics of DumbBot, the possible cause for this loss of performance may be the difference in the selection of the actions to perform.

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 (4.3).

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. In this scenario DipBlue should perform better than NoPress given it has more capabilities, however it performs worst than NoPress, decreasing the average position to 4.63. Since the only difference between both bots is the addition of the trust reasoning, a possible conclusion is that changing the trust ratios based on the attacks of the nearby opponents may not be an optimal strategy since it will increase aggressiveness towards those opponents. In fact, as analysed in Section 5.3.2, the player performs better the furthest its enemies are.

Scenarios 3 and 4 are used to assess how Naive and DipBlue act when in the presence of a Slave. As described in Section 4.4.2, a Slave may behave like a support player when allied to a player with the proper negotiation capacities, therefore 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 of 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, a DipBlue is paired with a Naive or another DipBlue, respectively, to measure the impact of betrayals and the way DipBlue detects them and reacts. In scenario 5, Naive has a worst performance than NoPress, Naive in scenario 3 and even than Slave when it was paired to DipBlue in scenario 4. However, DipBlue achieves the highest score in all 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 5.2 shows the average position of both DipBlues since there is no distinction between them. When paired with another instance of itself, DipBlue is able to detect betrayals and vulnerable to be detected betraying, therefore, when two instances of this bot are matched there is a high possibility 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, however, there is a decrease when compared to the performance of DipBlue in scenario 5. While in scenario 5 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 were created. Since in both scenarios all bots' performance 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. 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 behaviour as all players that have occupied that world power in previous games.

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.

According to the results obtained, hypothesis 3, which stated that a bot with communication performs better than a bot without communication, can be verified comparing the results of scenario 1 (bot without communication) and the results of bots that are not being used as a support player, such as Naive in scenario 3 and DipBlue in scenarios 4, 5 and 6. In all of these scenarios, bots using communication perform better than a bot without communication.

Hypothesis 4 affirms that a bot with trust reasoning has a better performance than a bot without it. 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 analysed. Scenarios 1 and 2 provide the information needed to reject the hypothesis for action-based trust reasoning, since NoPress in scenario 1 has a better performance than DipBlue in scenario 2. For negotiation-based, scenarios 5 and 6 can be considered. When using trust reasoning, DipBlue achieves a better performance in scenario 6 than Naive in scenario 5, for being able to detect and react to betrayals.

Scenarios 5 and 6 can also be used as a mean to validate hypothesis 5, which states that a bot loses performance if its betrayals are detected. This can be proven by comparing the results obtained by DipBlue in scenarios 5 and 6, since in scenario 5 DipBlue is able to betray without detection or repercussions and in scenario 6 its betrayals are detected, which results in the end of truce and alliances.

5.3.2 Correlation of Variables

In order to enhance the analysis of the obtained results, an inspection of dependencies between variables is needed, such as the impact of the number of years and the distance to allies and enemies in the resulting position of the bot. In order to better understand this dependency the correlation coefficient between the variables was calculated. By definition, the closer the correlation coefficient is to either -1 or 1, the stronger the correlation is between the two variables, where a coefficient of 1 is a direct correlation (when one variable increases the other increases) and a coefficient of -1 means an indirect correlation (when one variable increases the other decreases); a coefficient close to 0 means there is no dependency.

All correlation coefficients regard the player position, which represents the ranking of the player. it is 1 if the player ended in first place and 7 if the player ended in the last position. Therefore, negative coefficients mean the bigger the value of the variable the better the player's rank.

Figure 5.2 displays the inverse correlation coefficients regarding the data from all scenarios combined in order to collect overall information. The variables represent the years the game takes

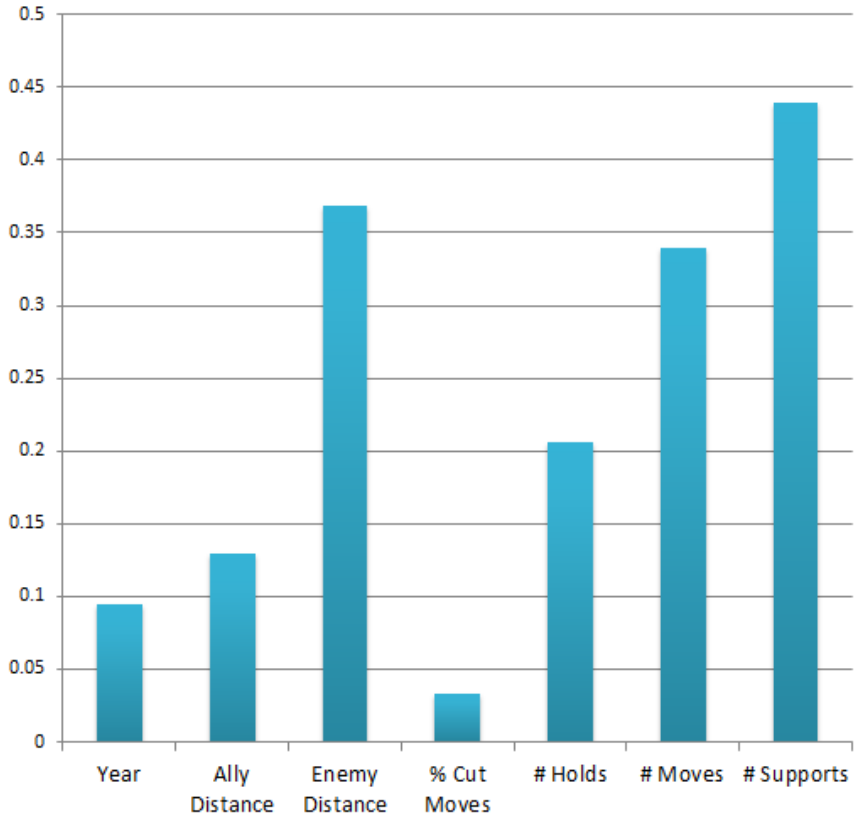

Figure 5.2: Inverse correlation with final position of the bot.

to end, distance to allies and enemies, percentage of moves cut and the number of holds, moves and supports. Since the correlation coefficients proved to be mostly negative, all values were inverted for better understanding and display purposes.

Hypothesis 1 states that the distance to the allies has a direct impact on the performance of the player, more specifically, the closer the allies the better the performance. In order to verify this hypothesis, there should exist a significant direct correlation between the average ally distance and final position, displaying negative values close to -1. In Figure 5.2 the correlation has a low positive value, which indicates that the correlation between the final position and the ally distance is the contrary of the expected and with very low values indicating the correlation is not significant.

Similarly, Hypothesis 2 states that the distance to enemies (opponents in a state of war) has an indirect impact on the performance of the player, i.e. the farther the enemies the better the performance. Figure 5.2 shows a relevant indirect correlation between the final position and the distance to enemies, which indicates that the distance to enemies does have a positive impact on the performance of the player.

The correlation of the final position with the years the game takes to end is very reduced

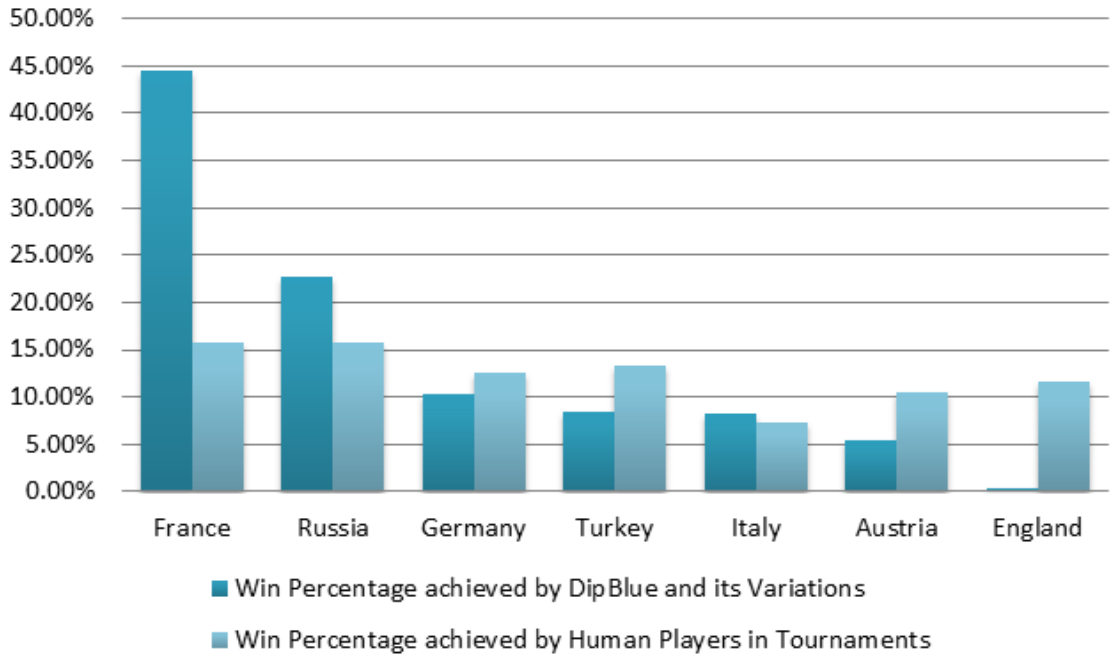

Figure 5.3: Average position of the bot for each power

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 cut moves: unlike what would be expected there is not a relevant impact of the effectiveness of the moves on the position of the bot.

Regarding the number of holds, moves and supports, they display a high correlation with the final position for the reason that when a player owns several units, these units will in turn perform several actions. In addition, if a player owns several units it means he owns at least the same amount of supply centres, 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. However, it is interesting to verify that the number of holds has the lowest coefficient of the three and that the number of supports has the highest coefficient. This difference is useful to better adjust the propensity to perform certain actions, meaning that to achieve a better performance, the bot should chose to perform supports over moves and moves over holds, by judging Figure 5.2.

5.3.3 Impact of Power

In games played by humans there is a slight difference in performance depending on the power the players are assigned to. To better understand the distribution of wins through the various powers, some studies were made using the results from tournament games. One study made by Eric Hunter shows a discrepancy between the powers [Hun14], which is illustrated in Figure 5.3. The results show an advantage of nearly double win percentage between France and Italy.

To better understand how the bots developed in this dissertation behave, the same experience was made and its results are presented in Figure 5.3 in comparison with the values obtained by Eric

Hunter. The bots display a higher disparity of values as compared to the tournament results. While the tournament win percentage range from 7.23% to 15.8%, the percentages of the bots ranged from 0.36% to 44.5%, which indicates that the bots have a different performance depending on the world power they are assigned to. Although the difference between the powers is much larger than in human games, the disposition of the wins throughout the powers is fairly the same, keeping France and Russia in first and second places, Germany and Turkey in the middle of the rank and finally Italy, Austria and England in the bottom of the table.

Given that the discrepancy is far greater than the existing asymmetry in tournaments played by human players, hypothesis 6 – which states that DipBlue should perform similarly regardless of the power it is assigned to – was rejected.

The results obtained in this experiment invalidate

5.4 Summary

The test scenarios had the purpose of enhancing certain aspects of the bots or their combination, producing results that allow to verify the validity of the proposed approach. Bots using negotiation proved to outperform other bots, while the usage of trust reasoning was proven essential when faced with the possibility of betrayals. The next chapter presents an overall summary and conclusions of this work regarding the bot and its strategies.

Chapter 6

Conclusions

This chapter concludes the dissertation and presents overall conclusions regarding the work made and how it fulfils the proposed objectives and answers to the hypotheses advanced in Chapter 1. It also presents the contributions to Diplomacy, DipGame and the general scientific area of MAS, followed by some guidelines of what could be done in the future, in order to further improve the bot developed in this dissertation.

6.1 Analysis of Results and Review of Objectives and Hypotheses

In the previous chapter, the several experiments conducted over a variety of test scenarios were described and analysed in detail. This section provides an overview of the results and how they relate with the proposed objectives and hypotheses, as proposed in Section 1.2.

Through the analysis of the results it is possible to conclude that the foundation of the bots, i.e. NoPress, has a worst performance as compared to the common opponent, DumbBot, even thought it was meant to be as close to the latter as possible. Since the objectives of this dissertation did not include the creation of an exceptional bot by the use of solution search and Diplomacy game tactics, the result achieved with NoPress was considered satisfactory, thus completing Objective 1. Objective 2 consisted on enabling bots to use L Language level 1, which was fully implemented and serves as the foundation for all negotiation in the bots developed in this work. The creation of successful negotiation tactics and the ability to detect and react to betrayals, as explained in Section 4.2, conclude Objectives 3 and 4. Finally, Objective 5 stated that the bots should be created using a flexible and scalable architecture, presented in Section 4.1, which is achieved through the used of Advisers (such as those proposed in Section 4.3) and the multiple adjustable parameters of the bot, obtaining archetypes such as the ones instantiated in Section 4.4.

Chapter 5 provides an analysis of the hypotheses. Hypotheses 1 and 2 are assessed in Section 5.3.2, Section5.3.1 provides a revision of hypotheses 3, 4 and 5 and hypothesis 6 is analysed in Section 5.3.3.

Overall, the results are in accordance with the expected and proposed, all the objectives were fulfilled and the fact that not all hypothesis were confirmed resulted in some interesting conclusions and new questions for possible future work.

6.2 Summary and Contributions

As seen before, Diplomacy offers a huge opportunity regarding the negotiation between the players. Although the game itself was built to enhance communication and negotiation and there are a lot of testbeds that support these aspects, most Diplomacy bots do not communicate at all and those that do cannot take advantage of the offered possibilities. The presented solution successfully takes advantage of negotiation, as an alternative to the traditional solution search approaches, in order to increase the performance of a simple bot. Negotiation is proven to be very powerful against innocent players and so are betrayals, thus reinforcing the need for trust reasoning.

The bot created through this dissertation is going to be provided to the DipGame community since there are no bots capable of communication available for testing and because DipBlue was created using a flexible and extendible architecture with the purpose of being easily used and reused by others. Along with the bots, the tools made to help improve their creation and testing, such as the Scout and the new game launcher, are also going to be provided. Additionally, the changes made directly to the contents of the DipGame library are going to be proposed as improvements to its developers to help improve the efficiency of development of future bots by the community.

Although the bot was created for the testbed DipGame, its contents and tactics may be applied to bots of other testbeds, such as DAIDE.

The obtained results can be extrapolated to other areas besides Diplomacy, since most techniques apply to the majority of MAS scenarios. Since these systems rely on communication between agents, the need for negotiation is essential to coordinate large groups of individual agents in order to achieve something unreachable if each agent acted alone. The impact of not fulfilling deals presented in this dissertation applies to several areas with consequences much more serious than in Diplomacy, such as MAS used in stock exchange or supply chain management.

6.3 Future Work

This section presents some future work that could improve the work developed in this dissertation. The impact of the improvements is uncertain without some previous testing, but most improvements seem promising, mainly the ones related to aspects tested and proved in other Diplomacy bots or similar environments.

6.3.1 Performance of Powers

When Diplomacy is played by humans, there is a slight unbalance of the win percentage depending on the power the players are assigned to. This effect was also observed with the bots developed, although with a higher discrepancy. Reducing this effect would be beneficial to achieve a more stable and robust player, capable of having a good performance without depending on the power it is assigned to.

6.3.2 Communication Capabilities

The negotiation strategies rely on communication by default. One of the most valuable improvements to be made is to increase the communication capabilities of the bot by allowing it to send messages in higher levels of the L Language (see Figure 2.5). After achieving this goal, the next step would be to adjust and improve the negotiation tactics to embrace the new capabilities and take advantage of them. Although not all existing bots are able to communicate in higher levels of the L Language, the simple act of launching two instances of the improved bot enables communication in the implementation level to exist between them. Currently, DipBlue uses L Language level 1. By analysing the L Language structure, an improvement to level 3 should result in a drastic improvement in the performance and abilities of the bot.

6.3.3 Reputation

Since the developed bot does trust reasoning, the next logical improvement is to enable the ability to use reputation. Although the two concepts are different, they usually appear together. Reputation is usually used as a way to estimate trust by sharing and inquiring information about someone or something. The usage of reputation could not only mean a way to gain information about the subject of the conversation, but also about the source of the information since the information can be inaccurate or simply a lie.

6.3.4 Optimization

Following the work of Dave de Jonge [Jon10], who's masters dissertation was the optimization of DumbBot [Nor13] with genetic algorithms, it is possible to apply the same concept to several values, attributes and coefficients of DipBlue in order to achieve an optimal configuration of the bot. The optimization can focus on the solution search heuristics, the negotiation, trust and communication tactics and the DipBlue Advisers weights and their own intrinsic parameters.

6.3.5 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 do each action during the game and to improve the bot's negotiation tactics. In addition, these learning capabilities can be used to do some prediction of the opponent's next moves during

the game. Together with Trust reasoning, the player would have a better way to determine its best alliances, which ones to keep and which ones to betray. 2

References

[App07] William J. Applegate, David L. and Bixby, Robert E. and Chvatal, Vasek and Cook. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, 2007.

[Bur11] Christopher Burnett. *Trust Assessment and Decision-Making in Dynamic Multi-Agent Systems*. PhD thesis, University of Aberdeen, 2011.

[Cal00] Allan B. Calhamer. *The Rules of Diplomacy*. Avalon Hill, 4th edition, 2000.

[Cal13] Allan B. Calhamer. The Invention of Diplomacy. http://www.diplom.org/~diparch/resources/calhamer/invention.htm, 2013. Accessed: 12-07-2013.

[Cam01] Murray Campbell. Deep Blue. Technical report, IBM, Yorktown Heights, NY 10598, USA, 2001.

[DAI13] DAIDE. DAIDE Homepage. http://www.daide.org.uk/w/index.php?title=Main_Page, 2013. Accessed: 15-07-2013.

[Dey10] Rui Jorge Gregório Deyllot. *Diplomacy – Base de Dados de Movimentos para Controlar Províncias*. Master thesis, Universidade de Aveiro, 2010.

[Dip13] DipAI. DipAI Homepage. http://games.groups.yahoo.com/group/dipai/, 2013. Accessed: 15-07-2013.

[dJS11] Dave de Jonge and Carles Sierra. Negotiation Based Branch & Bound and the Negotiating Salesmen Problem. In *Proceedings of the 14th International Conference of the Catalan Association for Artificial Intelligence*, Lleida, Catalonia, Spain, 2011.

[Fab13] Angela Fabregues. DipGame Homepage. http://www.dipgame.org/, 2013. Accessed: 12-17-2013.

[FS09] Angela Fabregues and Carles Sierra. A Testbed for Multiagent Systems Technical Report IIIA-TR-2009-09. Technical report, IIIA-CSIC, 2009.

[FS11] Angela Fabregues and Carles Sierra. DipGame: A challenging negotiation testbed. *Engineering Applications of Artificial Intelligence*, 24(7):1137–1146, October 2011.

[HL95] Michael R Hall and Daniel E Loeb. Thoughts on Programming a Diplomat. *Heuristic Programming in Artificial Intelligence*, 1995.

[Hun14] Eric Hunter. Solo Percentages. http://www.diplom.org/Zine/W2003A/Hunter/Solo-Percentages.html, 2014. Accessed: 04-01-2014.

REFERENCES

[IC13] IIIA-CSIC. IIIA Homepage. http://www.iiia.csic.es/, 2013. Accessed: 12-07-2013.

[JHr05] 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.

[Joh06] Stefan J. Johansson. On using multi-agent systems in playing board games. *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems - AAMAS '06*, page 569, 2006.

[Jon10] Dave De Jonge. *Optimizing a Diplomacy Bot Using Genetic Algorithms*. Master thesis, UAB, 2010.

[JS12] Dave De Jonge and Carles Sierra. Branch and Bound for Negotiations in Large Agreement Spaces (Extended Abstract). In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, pages 1415–1416, Valencia, Spain, 2012.

[KGL95] Sarit Kraus, Ramat Gan, and Daniel Lehmann. Designing and Building a Negotiating Automated Agent. *Computational Intelligence*, 11(972):132–171, 1995.

[Nor13] David Norman. David Norman's DumbBot. http://www.daide.org.uk/w/index.php?title=DumbBot_Algorithm, 2013. Accessed: 12-07-2013.

[otC13] Wizards of the Coast. Wizards of the Coast Homepage. http://company.wizards.com/, 2013. Accessed: 14-07-2013.

[PPG11] Sylwia Polberg, Marcin Paprzyck, and Maria Ganzha. Developing intelligent bots for the Diplomacy game. In *Computer Science and Information Systems*, pages 589–596, 2011.

[Rib08] João Santos Ribeiro. *DarkBlade - Um agente para Diplomacia*. Master thesis, Universidade de Aveiro, 2008.

[Rit03] Alan Ritchie. *Diplomacy — A.I.* Master thesis, Information Technology at The University of Glasgow, 2003.

[RNC+95] SJ Russell, P Norvig, JF Canny, JM Malik, and DD Edwards. *Artificial intelligence: a modern approach*. Prentice Hall, 3rd edition, 1995.

[Sar87] Daniel Lehmann Sarit Kraus. Diplomat, an Agent in a Multi Agent Environment: An Overview. Technical report, Leibniz Center for Research in Computer Science, 1987.

[SD13] Carles Sierra and John Debenham. Building Relationships with Trust. In *Agreement Technologies*, volume 8, chapter 29, pages 485–507. Springer Netherlands, 2013.

[Sha50] Claude E. Shannon. XXII . Programming a Computer for Playing Chess 1. *Philosophical Magazine, Ser.7*, 41(314), 1950.

[Sha13] Richard Sharp. The Game of Diplomacy. http://www.diplomacy-archive.com/resources/god/one.htm, 2013. Accessed: 12-07-2013.

[vH13] Jason van Hal. Jason van Hal's Homepage. https://sites.google.com/site/diplomacyai/home, 2013. Accessed: 15-07-2013.

REFERENCES

[WCW08] Adam Webb, Jason Chin, and Thomas Wilkins. Automated negotiation in the game
2 of diplomacy. Technical report, Imperial College London, 2008.