TÉCNICO LISBOA

AI Player for board game Diplomacy

Sancho Fernandes de Mascarenhas

Thesis to obtain the Master of Science Degree in

Computer Science Engineering

Supervisor: Prof. Pedro Alexandre Simões dos Santos

Examination Committee

Chairperson:	Prof. José Carlos Martins Delgado
Supervisor:	Prof. Pedro Alexandre Simões dos Santos
Member of the Committee:	Dr. Samuel Francisco Mascarenhas

May 2017

Dedicated to Gonçalo Henrique Santos Pires do Amaral Xavier

Acknowledgments

The strength and help that has been given to me during the writing of this dissertation were indispensable to its conclusion. For that, I would like to write a few words of appreciation.

First, to my family, father, mother, brother, sister and grandmother who always supported me and gave me the opportunity to get to where I am and be who I am. A special thanks to my sister, Vera, who dedicated some of her time to this dissertation. Second, to my supervisor Pedro Santos for his good orientation and patience during the elaboration of this dissertation. I would also would like to thank mr. John Newbury for his patience and explanations about the DAIDE environment. Lastly, a big thanks to my friends for the good times I spent with them during the breaks of this dissertation, for cheering me up and for the concern they showed about the evolution of this work, even when I didn't want to talk about it.

Resumo

O Diplomacy é um jogo de estratégia onde 2 a 7 jogadores competem pela sua supremacia sobre a Europa no virar do século XX. É um jogo onde a sorte ou a aleatoriedade não tem qualquer papel e os jogadores negoceiam entre si de forma a ganharem vantagem. O tabuleiro está dividido em 75 províncias onde 34 são chamados "supply centers". Os turnos estão divididos em fases dedicadas à negociação, aos movimentos das peças dos jogadores e a ajustes do tabuleiro. No final de cada ano/turno, cada jogador ajusta o número de peças ao de "supply centers" que possui. Ganha o jogador que, no final de uma estação, detenha 18 dos 34 "supply centers" disponíveis no tabuleiro.

Neste trabalho foi criado um jogador automático (bot) dedicado ao jogo acima apresentado. O Tagus é um bot desenvolvido para a plataforma de jogos de Diplomacy oferecida pelo projecto DAIDE, usando um kit de desenvolvimento oferecido pelo mesmo. Tirando partido da negociação que faz com os restantes jogadores e de uma biblioteca de aberturas, o bot ganha vantagem a nível do jogo, sempre com o objectivo de ganhar. O tipo negociação feita é simples e inclui apenas propostas de aliança ou paz com base na tensão e confiança com os outros jogadores. As experiências feitas revelam que negociação traz vantagens às partes intervenientes. Além disso, tornou-se claro que, em alguns casos, o uso de bibliotecas de abertura fortalece os seus utilizadores no início do jogo.

Palavras-chave: Diplomacy, Bibliotecas de abertura, Negociação, Inteligência Artificial, Jogador Automático, DAIDE, Sistemas Multi-Agente

Abstract

Diplomacy is a strategy board game in which 2 to 7 players compete for their supremacy over Europe in the turn of the 20th century. It is a game where luck or randomness has no role and players negotiate with each other in order to gain advantage. The board is divided into 75 provinces where 34 are called supply centers. Each turn is divided in phases dedicated to negotiation, the movement of pieces and board adjustments. At the end of each turn, each player player adjusts the number of pieces to the number of supply centers they have. The player who, by the end of a season, detains 18 out of the 34 supply centers available in the board wins the game.

In this work, it was created an automated player (bot) dedicated to play the game presented above. Tagus is a bot developed for DAIDE's game platform, using the offered development kit. By negotiating with other players and using the opening libraries built in it, the bot gains advantage in the game, always with the goal of winning. The type of negotiation is simple which included only peace treaties and alliances based upon the tension and trust with other players. The experiments made reveal that negotiation brings advantage to the involved players. It was also clear that, in some cases, the opening libraries strengthen its users at the start of the game.

Keywords: Diplomacy, Negotiation, Opening Libraries, Artificial Intelligence, Automated Player, DAIDE, Multi-Agent System

Contents

Acknowledgments v
Resumo vii
Abstract ix
List of Tables xv
List of Figures xvii

Glossary **xix**
Glossary xix

1 Introduction **1**
1.1 Motivation 2
1.2 Goals 3

2 Related Work **5**
2.1 Negotiation 5
2.1.1 Automated Negotiation 5
2.1.2 Automated Negotiation Models 6
2.1.3 Negotiation Protocols 6
2.2 Diplomacy: The Game 7
2.3 Diplomacy computational complexity 10
2.4 Negotiation Testbeds 10
2.4.1 MAGNET 10
2.4.2 TAC 11
2.4.3 ART 11
2.4.4 CT 11
2.4.5 GENIUS 11
2.4.6 Diplomacy Testbeds 11
2.4.7 Chosen testbed 15
2.5 State-of-the-art 15
2.5.1 Diplomacy bots 15
2.5.2 Conclusion 20

3 Tagus Bot's Architecture and Implementation **21**
3.1 Tagus . . . 21
3.2 Military Command . . . 22
3.2.1 Province evaluation . . . 22
3.2.2 Betrayal . . . 23
3.2.3 Determinism vs Non-Determinism . . . 23
3.3 Opening Library . . . 23
3.4 Foreign Relations . . . 24
3.4.1 Trust Factor . . . 24
3.4.2 Tension Factor . . . 24
3.5 Breaking Agreements . . . 25
3.6 Arquitecture at work . . . 25
3.7 Implementation . . . 28
3.7.1 Installing DAIDE . . . 28
3.7.2 Bot Implementation . . . 28

4 Experiments **31**
4.1 Methodology . . . 31
4.2 Automation . . . 32
4.3 Test Series . . . 32
4.3.1 Series 1: 6 RandBot and 1 Tagus . . . 33
4.3.2 Series 2: 6 DumbBot and 1 Tagus . . . 33
4.3.3 Series 3: 4 DumbBot and 3 Tagus . . . 33
4.3.4 Series 4: 4 BlabBot and 3 Tagus . . . 34
4.3.5 Series 5: 4 Albert and 3 Tagus . . . 34

5 Results **35**
5.1 Series 1: 6 RandBot and 1 Tagus . . . 35
5.2 Series 2: 6 DumbBot and 1 Tagus . . . 37
5.3 Series 3: 4 DumbBot and 3 Tagus . . . 38
5.4 Series 4: 4 BlabBot and 3 Tagus . . . 39
5.5 Series 5: 4 Albert and 3 Tagus . . . 41
5.6 Summary . . . 42

6 Conclusions **45**
6.1 Achievements . . . 45
6.1.1 Subgoal #1 . . . 45
6.1.2 Subgoal #2 . . . 45
6.1.3 Subgoal #3 . . . 46
6.2 Theoretical conclusions . . . 46

6.2.1 Diplomacy and Automated Negotiation 46
6.2.2 Tagus' Negotiation Protocol 46
6.3 Future Work 47
6.3.1 Press Level 47
6.3.2 Opening Libraries 47
6.3.3 Negotiation 48
6.3.4 New test series 48

Bibliography **51**

List of Tables

2.1 Press Levels in DAIDE 12

5.1 Results of series 1 with: number of games per power, number of wins per power, percentage of wins of games played per power 35
5.2 Results of series 2 and comparison between the bots. 37
5.3 Results of series 3 and comparison between the bots. 38
5.4 Results of series 4 and comparison between the bots. 40
5.5 Results of series 5 and comparison between the bots. 41

List of Figures

2.1 Standard Diplomacy game map. 8
2.2 Example of a standoff. 9
2.3 Example of a supported attack and its result. 9
2.4 Dipgame's L Language table. 15

3.1 Tagus' architecture diagram 22
3.2 Tagus loads the opening library 26
3.3 Tagus updates the trust and tension factors. 26
3.4 Tagus ponders defensive peace or alliances. 26
3.5 Tagus ponders offensive alliances. 27
3.6 Tagus calculates the value for each province. 27

5.1 Series 1: average supply center count per year per power. 36
5.2 Series 1: number of turns per game. 36
5.3 Series 2: average supply center count per year per power. 37
5.4 Series 2: number of turns per game. 38
5.5 Series 3: average supply center count per year per power. 39
5.6 Series 3: number of turns per game. 39
5.7 Series 4: average supply center count per year per power. 40
5.8 Series 4: number of turns per game. 41
5.9 Series 5: average supply center count per year per power. 42
5.10 Series 5: number of turns per game. 42

Glossary

AI Artificial Intelligence. xvii, 1, 5, 16, 17

AN Automated Negotiation. xvii

DAIDE Diplomacy AI Development Environment. xvii

KMB KissMyBot. xvii

MAS Multi-Agent System. xvii, 2, 5, 10

Chapter 1

Introduction

Artificial Intelligence (AI) is a research field which, since its beginning, tries to understand the nature of thoughts and intelligence[1]. More recently, through computer science, the main goal has been to create computers provided with intelligent behavior and capable of doing tasks that humans and animals would do. This goal has not been easy to accomplish because of the many difficulties that arise, e.g., a wide ability to communicate. Despite the evolution of the field of Artificial Intelligence and the potential to solve many problems, others have emerged. For instance, an AI may be capable of simple conversations with a human, but for it to evolve to a higher level of communication, it needs a higher ability of processing and understanding.

Presently, it is possible, to create a computer capable of solving a calculation problem in a matter of seconds that a human being would need days, weeks or even years, while using paper and pen. . However, there are other tasks or problems that a computer has troubles or is not capable at all of solving them. A simple example is to maintain a cohesive and linked dialog. Other problems that AI tries to solve include reasoning, accumulation of knowledge and its representation, planning and natural language processing. An area where the solutions developed by AI are widely suited is the area of games. The creation of intelligent automatic players for the different games requires many planning, solution search and, more recently, natural language processing techniques. Search algorithms became very effective and widely used in this field as games like chess, checkers, GO and other zero-sum games turned out popular for the AI community. Although it is of the scientific community's interest, players, through the years, also demanded more intelligent, reliable and believable non-playable characters (NPC) to improve their gaming experience.

An example of a game, without considering other games such as chess or pong, in which players could sense intelligent agents was *PAC-MAN* [1]. In this game, the player controls a round shaped character through a labyrinth like map. In the same map, there are four ghosts that either chase the player or run away from him/her. The ghosts use state machines that define in what mode the ghosts are in. While in the chasing mode, each ghost had a different chance of either chasing the player or choosing a random direction. The same idea applies while they are running away from the player. Until the mid 90s,

[1] Midway Games West, Inc., 1979

other game AIs were not that much more sophisticated than *PAC-MAN*'s ghosts [1]. Despite all the techniques and solutions that have emerged, sometimes, they are not always as good as required for some games played in daily life. This is mainly because, many games have a wide solution space and the optimal solution is found in a deep level of the search tree, i.e., the set of possible plays. Thus, exploring the search tree until the solution is found isn't always possible. For this reason, the use of heuristics (an approach to problem solving, learning, or discovery that employs a practical method not guaranteed to be optimal or perfect, but sufficient for the immediate goals) gained some importance. Nevertheless, some efforts have been made lately to improve search algorithms, such as the Monte Carlo Tree Search (MCTS). This algorithm has been showing positive results in the area of games, more specifically in Go [2]. As of 2016, a program named AlphaGo, developed by Google Deepmind, bested a professional Go player. AlphaGo uses the MCTS via neural networks [3].

Nevertheless, for some games, the improvements mentioned above are not enough. Some games have other relevant information that either extend the search tree further or has a different impact in the game itself. For instance, a game that involves negotiation, or dialog or any other type of human interaction is greatly influenced by the outcome of that same interaction. An example is multi-player games seen as a Multi-Agent System (MAS).

In light of what was described above, this dissertation will focus on a board game in which it is not trivial for search algorithms to play well. This game can be seen as a MAS and is called *Diplomacy* [2].

1.1 Motivation

Since the primordial times of our civilization, negotiation has been a major type of interaction between human beings. The necessity of settling differences became more and more essential. These differences could be related to trades for example, where two or more parts try to reach an agreement over some sort of goods. At some point, for some reason, everyone is forced to negotiate over something no matter its nature. Also, negotiation is the most flexible and efficient method of overcoming different matters. Currently, negotiation takes an important role in Multi-Agent Systems. As the name suggests, a MAS involves multiple agents in an environment, each with a goal. Sometimes, the goal of an agent is not compatible with the goal of another one and conflicts between these agents are, normally, unavoidable. These conflicts include, for example, resource management, where resources can be of any type (human resources, raw materials, finances, etc.), depending on the context, agents compete against each other for a certain good. An example of a multiple-agent system, where resource management is of concern, is the board game Diplomacy where players compete for territories to expand their armies.

Diplomacy is a board game created by Allan Calhamer in 1954 and it was playtested and tuned until 1958, and since then many companies have published it[4]. With the arrival of the internet, Diplomacy also started being played via e-mail and later in 1984, the first Diplomacy video game was released.

The main characteristic of Diplomacy that makes it so interesting to the scientific community is the fact that it includes no luck, no random factors and there is no hidden information on the board (for

[2][Allan B. Calhamer, 1954

example, there are no cards) other than the agreements made between players as the game is played. The main component of this game is negotiation and persuasion. Players make deals with each other in order to satisfy their interests and are free to break those deals anytime they see fit. These deals may include, for example, alliances, peace treaties or information exchange. Negotiations are done in a specific phase of the game referred to as Diplomacy period. Also, during this phase, players are allowed to make their deals in different rooms, so that the remaining players are kept out. Thus, the only information considered hidden in this game are the deals between players.

Comparing to other zero-sum games, such as chess, the negotiation factor in Diplomacy as well as the number of players and units make its branching factor, i.e., the number of possible plays in one turn, larger than the other multiplayer games. This means that it is almost impossible to get the optimal solution. Even with the above mentioned MCTS, the solution would not be trivial to find. Also, the non-determinism of Diplomacy make the use of heuristics not very effective. Furthermore, it is agreed by the scientific and gamer community that there is no heuristic that can evaluate with precision the state of the game [5]. Players only have access to the information the board offers and their own deals, which are hidden and are not accessible to everyone.

Overall, Diplomacy's seems a rather simple game but in theory it can be quite complex. This is the reason why it is an interesting target of study in negotiation and artificial intelligence.

1.2 Goals

Considering the motivation described in the previous section, this dissertation focuses on the board game Diplomacy. As this game offers a good environment to test negotiation, the main goal of this project will be to build an automated player, i.e, an artificial intelligence (or bot) capable of playing Diplomacy including the taking advantage of negotiation. The player will make use of the tension created between oneself and other players to decide what, when and with whom will he/she negotiate with. Besides negotiation, the bot will also take advantage of strategic moves in an initial phase of the game in order to be strong right from the start. Thus, this project can be divided in the following subgoals:

Subgoal #1 Create a functional automated player for board game Diplomacy;

Subgoal #2 Elaborate strategic moves that the player can follow to be stronger in the start of the game;

Subgoal #3 Enhance the player with negotiation abilities to gain advantage over the others as the game progresses;

Chapter 2

Related Work

2.1 Negotiation

Negotiation is the target of many studies in different areas such as economics, international relations, social sciences and more [6]. Some of those studies were made by Lewicki and can be consulted in [7].

Since the beginning of Humanity, human beings have needed to overcome problems that involve two or more entities, e.g., the struggle over a certain territory. In order to solve said problems all parts involved try to make a balanced deal that pleases everyone. In other words, people **negotiate** and the goal is to settle differences between the involving parts. There are several reasons that lead two or more parts to negotiate: business, where parts negotiate the purchase and sell of goods and services, diplomacy, where peace accords are established, etc.

The area that is of interest for this thesis is the Artificial Intelligence.

Presently, the demand for computational agents capable of reaching agreements on the behalf of their owners is increasing [6]. Agents refer to entities that act in a certain environment in order to achieve its goals. The branch in AI dealing with negotiation is called **Automated Negotiation** (AN) and focuses on negotiation in MAS, where two or more agents try to settle a deal in order to achieve desirable outcomes.

2.1.1 Automated Negotiation

In Automated Negotiation, it is assumed that agents make agreements and these agreements belong to a set called the agreement space. Agreements from this set are proposed and either accepted or rejected by the agents. Once the agreement is accepted, a commitment is made between the agreeing parts [8]. In automated negotiation, the agents taking part in the negotiation try to reach this desirable outcome by maximizing an utility, i.e., a particular purpose, of their own. Therefore, the agents should try to develop an agreement that satisfies, to some extent, its utility and that its benefit to other agents as well.

Although it is stated that “a commitment is made between the agreeing parts”, in the board game Diplomacy, the agreeing parts do not always honour the commitment and can betray each other. On

the other hand, it is not clear if Diplomacy can be included in automated negotiation regarding the commitment aspect when one considers the whole game. However when one isolates each round of negotiation, it is possible to include Diplomacy in the automated negotiation category.

2.1.2 Automated Negotiation Models

There are several models of automated negotiation. In [9], three models are adopted by the authors: 1)game-theoretic model, 2) heuristic model and 3)argumentation-based model. The game-theoretic model analyses specific negotiation situations and its results regarding the strategy each negotiator should follow. It basically looks to maximize the negotiation outcome. The second model, the heuristic model, offers ways to assist the negotiators and strategies in order to move towards an agreement and get a good result (not necessarily the optimal one). In both these models, negotiation is made through an iterative process of proposals and counter proposals are exchanged. The third model, argumentation-based model, lets the negotiators argue and justify their negotiation position and persuade others. In this last model, besides the proposals and counter proposals, the negotiators also exchange other types of messages, such as promises and threats.

Later studies, such as in [6], present different models of automated negotiation. These are: the theoretical model and the computational model. The theoretical model consists on the description, specification and reasoning of key features of the agents. Furthermore, this model focus on the outcome of the negotiation process and not on the process itself. The process is mainly viewed as a mean to select a solution from a set of candidates. On the other hand, the computational model specifies key data structures of an agent and the way these structures are operated. This model is successfully used in real-world domains. Contrary to the theoretical model, the computational model focus on both the outcome and the negotiation process itself. It also follows a set of more realistic assumptions compatible with the real world for which the theoretical models do not, e.g., the rational capability of the agents. Comparing the models adopted in [6] and the ones in [9], we can say that the theoretical model is similar to the game-theoretic model while both heuristic and argumentation-based models are similar to the computational model.

2.1.3 Negotiation Protocols

Another way to classify the type of negotiation is by looking at its protocol. A two party negotiation is called bilateral negotiation while a three or more party negotiation is called multilateral negotiation. A good example of multilateral negotiation is an auction house, where several(usually more than two) parties (bidders) negotiate over a good. Although there are multiple bidders in multilateral protocols, the item or good that is being bid and its price is the same for everyone. On the other hand, bilateral negotiations consists of two entities offering proposals in an alternating way until both agree or simply give up on the negotiation. Regarding bilateral negotiation we can identify a variant called multiple bilateral negotiation. As the name suggests, in this protocol, there are several bilateral negotiation occurring simultaneously. One agent can, at the same time, negotiate with two or more agents and use

different strategies with each one of them. This protocol is very common in the real world as an entity, for example a supplier of some material negotiates the price of a good, most of the time independently, with several other entities.

Regarding automated negotiation, multiple bilateral negotiation has some problems precisely because the outcome of several negotiations affect one common agent. As a result, some inconsistencies between agreements can occur. Until today, some works have tried different approaches to overcome this issue. As example is the work by [10] to overcome the problem by allowing the negotiator to switch between protocols. Another interesting work is the one described in [11] which makes use of sequential bilateral negotiations in order to approximate to multiple bilateral negotiations. The Diplomacy board game, uses the multilateral protocol as players can negotiate with several others simultaneously.

2.2 Diplomacy: The Game

In Diplomacy, preferably, 7 players control a European power set in the turn of the 20th century. The rules presented below were extracted from Diplomacy game's rule book [12]

These powers include Russia, Turkey, Austria, Italy, Germany, England and France and each powers compete against the others to establish supremacy over the continent. Also, the map includes 75 provinces where 34 are supply centres. Some supply centres are called "home" supply centres as they are controlled by the specific power even when it is not occupied. For example, the home supply centres of Russia are are Moscow, St. Petersburg, Warsaw and Sevastopol and are controlled by Russian even if no Russian unit occupies it. However, it stops being controlled, when another power (e.g. Turkey, Austria) occupies it. When a player controls 18 of the 34 supply centres, he/she wins the game. In order to control these provinces, each player possess armies and fleets that he/she must maneuver throughout the map.

Each round is composed of 3 distinct phases: 1) the diplomacy phase, 2) the writing orders phase and 3) the execution of the written orders. A round is defined by a season (Spring or Autumn) and a year. For example, the game starts in Spring of 1901. Once all 3 phases are completed, it is said that a campaign (the round) is complete and a new one begins. Looking back to the example given before, when Spring of 1901 campaign ends, Autumn of 1901 campaign would begin. Autumn is the last season of each year, which means that once Autumn is over, the next round would be Spring of the next year. At the end of each year (end of Autumn), each player must adjust his/hers playing pieces (from here on referred to as *unit*) accordingly to the number of supply centres he/she controls. There are some variants, like the one presented in [5], that consider five seasons, Spring, Summer, Fall, Autumn and Winter. In these variants, diplomacy and order phases occur in Spring and Fall, conflict resolution is in Summer and Autumn and the adjustments are made in Winter. In the end, the rules are the same, only the organization changes.

The diplomacy phase is the time when players are allowed to negotiate and bargain whatever deal they desire. However, no agreement is binding: each player may or may not choose to later betray the players he/she made a deal with. This happens before the orders for each unit is written. Also this phase

is normally timed and, in casual play, it is up to the players to decide how long this phase will take. After diplomacy is made, the orders for the units to take must be written. Each player writes what each of his/hers controlled units will do. Once written, the orders cannot be altered. This phase is also timed. After the order have been written, they are passed to the player next to the owner of the orders and are read to all players. The orders have to be written in a specific format: they must indicate the round they will be performed in, the unit itself (army or fleet), the province the unit is occupying, the action and the province which the action will take place. This format ensures that the orders are clear and not ambiguous.

A unit (army or fleet) has a set of orders it can perform, and one order can only be performed per round. These orders are: move, support, stand or hold and convoy. The simplest order is the hold order. It is the order given, by default, to units that have no order assigned to them or it can actually be written down. The unit ordered to stand stays in the province it is occupying. The move order allows units to move to adjacent provinces. But there are some exceptions: armies cannot move to sea provinces but fleets can move to sea or coastal provinces. Also, there are areas bordering two different seas. In this case, the border used by a fleet to occupy the area of interest is the **only** border from which the fleet can move out. The support move allows a unit to help another unit, controlled by the same power or not and to either defend its position or move to another area. This is very relevant because every unit, disregarding its type, has the same force. So in case of conflict, only the number of units are considered and not their types. Finally is the convoy order. The convoy order allows a power to move an army

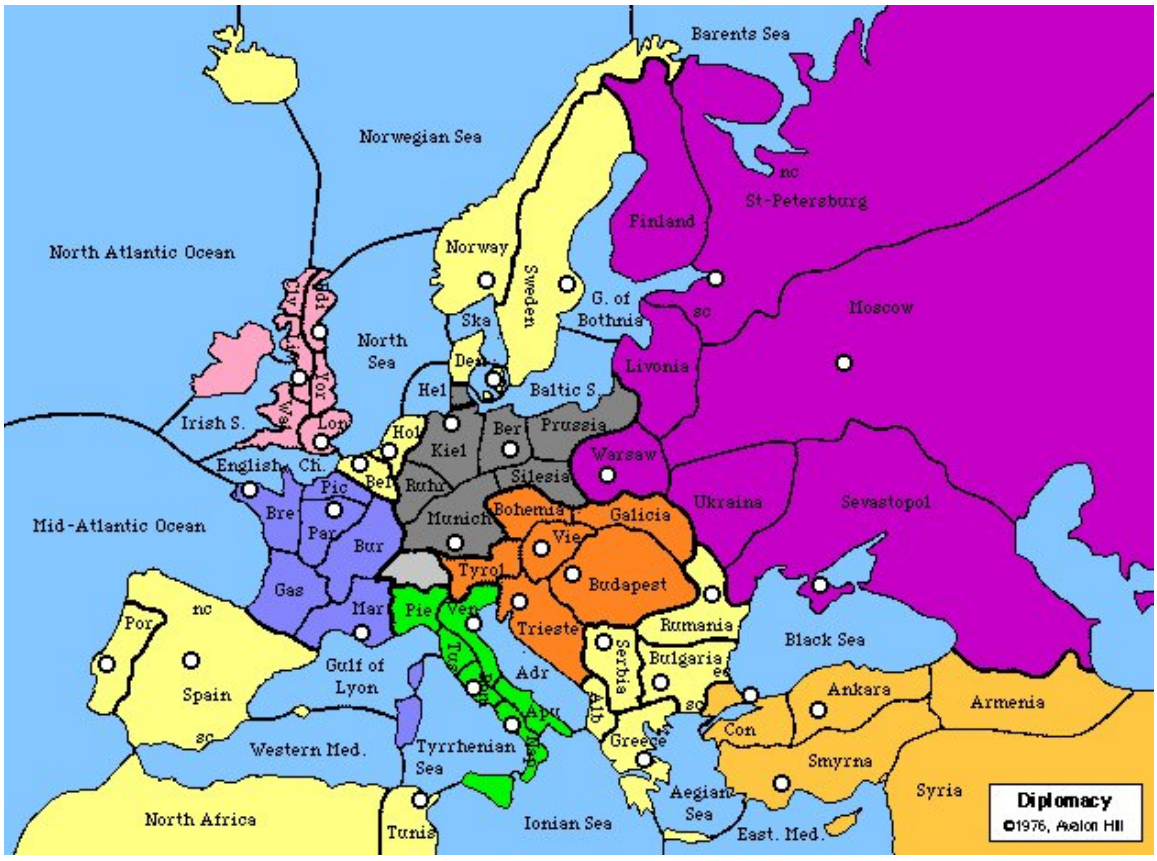

Figure 2.1: Standard Diplomacy game map.

across a sea area. This can only be done if the sea area to be crossed is occupied by a fleet, but not necessarily controlled by the same power. It is possible for an army unit to be convoyed across a set of seas in case these seas are occupied by fleets and if all the fleets are ordered to convoy the same army.

Finally, as the orders are read to all players, the different units are moved to the borders according to their assigned order. All orders are then executed **simultaneously**. When executing the orders, conflicts may occur and which must be resolved. A conflict occurs when two units move to or try to occupy the same region that only one unit may occupy it. This is the situation where the support order becomes relevant as it strengthens the supported side. For example, if a French army from Picardy and a German army from Ruhr both try to occupy Belgium, this will result in a standoff and both armies stay in their initial position (Figure 2.2).

However, if the army from Picardy had support from an army from Burgundy, the army from Picardy would be victorious and would be able move to Belgium whilst the Ruhr army would stay in Ruhr. The Burgundy army would also stay in its initial position (Figure 2.3). This was a very simple but common event in Diplomacy, however there are many other types of conflict that would not result in such a trivial

Figure 2.2: Example of a standoff.

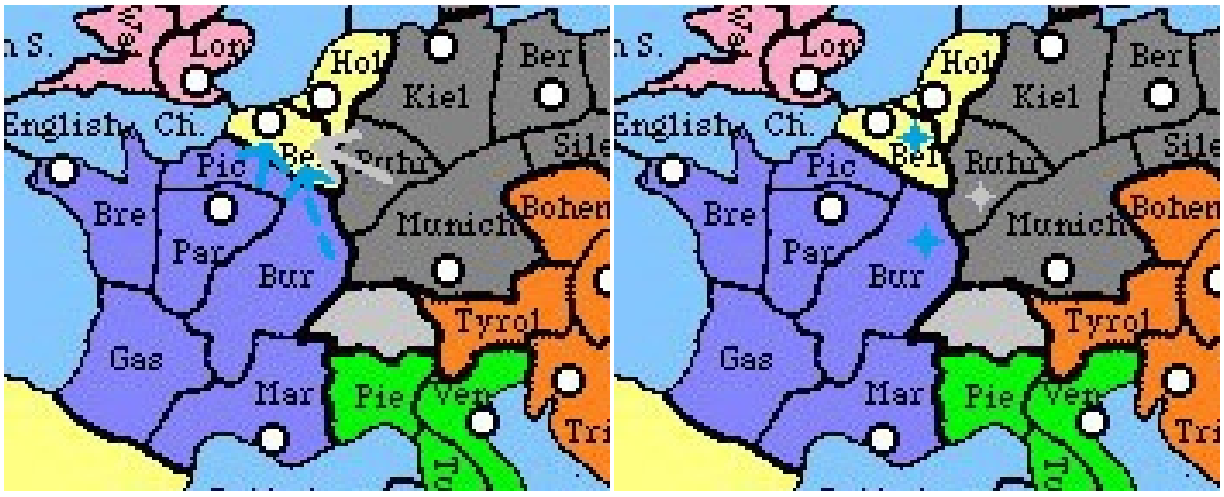

Figure 2.3: Example of a supported attack and its result.

outcome and will not be presented in this thesis.

When the Autumn campaign is finished, there is a Build/Disband phase in which players must adjust the number of units in such a way that in the end it equals the number of supply centres they control **in that moment**. When the number of supply center is greater than the number of units, the player adds units until the numbers match. The new units are placed in the **home supply centres if they are not controlled by another power**. If the number of units controlled is greater than the number of supply centres owned, the player has to remove enough units so that both numbers match.

2.3 Diplomacy computational complexity

Comparing to other games, Diplomacy is considered computationally more complex then Chess, Checkers or Go. According to [13], Chess has a branching factor of around 30 and depths between 7-12 while the game Draughts has a 10-15 branching factor due to the fact that each piece has less moves available per turn. In a 19x19 board of Go, a player has 361 options as the first play of the game. This number does not change much: once the players reach midgame, some ”stones” get removed, increasing the number of possible plays for the next turn. Fabregues [14] states that the branching factor of Go is about 200. Regarding its depth, it reaches a depth of 150 turns.

On the other hand, considering a Diplomacy game board with 34 pieces (when all supply centres are occupied) and each piece has an average of 5 moves, there are about 45 million available moves each turn, i.e, a branching factor of 45 million. Worst case scenario, more moves can be considered, up to a total of 10 moves, resulting in a branching factor of $2 * 10^{15}$.

2.4 Negotiation Testbeds

Regarding multi agent systems, there are several testbeds, a platform for testing, that allow automated negotiation experimentation. Some testbeds for MAS include Multi AGent NEgotation Testbed, MAGNET for short[15], Trading Agent Competition, TAC[16], The Agent Reputation and Trust Testbed, ART [17], the Colored Trails Game testbed, CT[18], and the Generic Environment for Negotiation with Intelligent multipurpose Usage Simulation, GENIUS[19]. Each of these platforms has its own characteristics and specifications which will not presented in detail in this dissertation.

2.4.1 MAGNET

The Multi AGent NEgotation Testbed testbed is a simple but generalized environment that allows multi agent contract negotiationby the use of auctions. It has a generalized market architecture that supports several transaction types from simple buying and selling to complex multi-agent negotiations. The architecture is organized into three components: the exchange, the market and the session. Each of these components is analyzed with detail in [15].

2.4.2 TAC

In Trading Agent Competition agents take the role of travel managers. As so, their goal is to come up with the best travel package for the client, keeping into account the utilities of the client. TAC is also prepared to be used for an annual competition.

2.4.3 ART

Agent Reputation and Trust Testbed was built with the goal of establishing a platform that can test the trust worthiness and reputation of the agents. In short, the agents are asked by the client for appraisals of a certain painting. The agent must decide either to ask for make the appraisals in exchange of a fee or to deal with the appraisal itself. The higher the agents reputation, better are the chances other agents pay it.

2.4.4 CT

Colored Trails Game Testbed focused on decision-making. The game in this testbed consists of exchanging coloured chips between agents in order to move around a coloured square board. Negotiation in this game includes committing or retracting to proposals, trading chips with other players and planning towards a certain goal.

2.4.5 GENIUS

The goal of the Generic Environment for Negotiation with Intelligent multipurpose Usage Simulation is to facilitate the design of negotiation strategies. It does not have a static Domain and assumes that the given domain establishes a number of challenges and a range of values. All agents know such domain and negotiation are a mapping between the challenges and their possible values.

2.4.6 Diplomacy Testbeds

Initially, Diplomacy had a few platforms where players could play against each other or against low capability bots. As an example, the community has the digital version released by Avalon Hill in 1984 in which the “computer opponents are not fully intelligent” [20]. With the increase in popularity of the game among players, and equally by the scientific community, some testbeds created especially for Diplomacy started to appear and being the ones that will be discussed in more detail below. I will focus in two particular testbeds: the Diplomacy AI Development Environment (DAIDE) and the Dipgame.

2.4.6.1 DAIDE

The Diplomacy AI Development Environment, as the name clearly states, is and environment created in January 2002 by a group of programmers with the purpose of allowing several Diplomacy bots to compete against each other. This environment consists of a communication model, a communication

protocol and a specific language, including syntax and semantics in which negotiations and instructions are expressed. It also provides some useful utilities such as an arbiter that serves as a judge, several libraries that facilitate the creation of new bots and a diversity of bots to test new AIs[1]. This platform (server and client) was developed in C/C++ language.

In order to categorize the communication capabilities of the bots, the group of DAIDE created several levels of communication called Press levels. The higher the level, the more complex is the communication capability of the bot. There are 15 levels but the highest level reached so far is the 4th level (e.g., Level 30 Multi-part Offers in Table 2.1).

Table 2.1: Press Levels in DAIDE

Level	Category
Level 0	No Press
Level 10	Peace and Alliances
Level 20	Order Proposals
Level 30	Multi-part Offers
Level 40	Sharing out the Supply Centers
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

The infrastructure system of DAIDE, is set based on a client-server communication model. The server part runs the game and receives the orders from all players. It is also the server that runs the adjudicator algorithm responsible for the generation of the next turn depending on the results of the orders of the player. Therefore, the players never communicate with each other directly. The system supports two types of clients, the player type and the observer type. The latter, and as the name states, only observes the state of the game and it does not send any order to the server. By using a TCP/IP protocol, it is possible to set a game where the client and the server are not in the same machine.

DAIDE follows a particular syntax regarding message exchange. It uses a variety of tokens to identify the type of message that is being exchanged. To demonstrate how the messages are built without entering into much detail, press messages are as follows:

- FRM (power) (power power . . .) (press_message)
- FRM (power) (power power . . .) (reply)

where FRM is the token used to identify received press messages. (power), which can be AUS, ENG, FRA, GER, ITA, RUS or TUR, identifies which power sent the message. (power power . . .) is a list, with at least one element, containing all the receiving powers, (press_message) is the actual press message

[1] Introduction to DAIDE webpage: http://www.daide.org.uk/index.php?title=Introduction

and (reply) is a reply to a previously sent press message. The (press_message) has three different forms:

- PRP (arrangement)
- TRY
- HUH

(arrangement) can be:

- PCE (power power . . .) for peace proposals
- ALY (power power . . .) VSS (power power . . .) for alliance proposals or
- DRW to propose a draw between the remaining players
- TRY or HUH when an error was detected in a previous message

Finally, (reply) can be:

- YES (press_message) to accept a proposal,
- REJ (press_message) to refuse a proposal
- BWX (press_message) to neither accept or reject (regarded as “none of your business”)
- HUH (press_message) for errors[2]

An example of press message can be:

- PRP (ALY (RUS AUS) VSS (TUR))

In this example, Russia proposes Austria an alliance against Turkey. If it is accepted, it is followed by the reply:

- FRM (AUS) (RUS) (YES (PRP (ALY (RUS AUS) VSS (TUR))))

A rejection would be:

- FRM (AUS) (RUS) (REJ (PRP (ALY (RUS AUS) VSS (TUR))))

There are other important tokens in the syntax of DAIDE used for the communication between the client and server. Some examples are:

- TME for information on the time of each phase;
- SCO token for information about supply centre ownerships;
- HLO token which defines which power the client will be playing;

[2]For more info on the syntax of DAIDE, http://www.ellought.demon.co.uk/dipai/daide_syntax.pdf

- ORD token for information about an order and its result;
- MDF token for information on the map;
- NOW token indicates the current turn, and the current unit positions;

It is worth mentioning that in DAIDE it is played the five seasons version of the game.

DAIDE is a very well documented environment and has its own discussion group which can be contacted in the DipAI Yahoo! group.

2.4.6.2 David Norman's Development Kit

David Norman's Development kit, named AiClient Framework, later updated by John Newbury, forms the basis of the C++ bots (clients) of DAIDE. Before the update by John Newbury, the AiClient Framework used the Microsoft Foundation Classes (MFC). The MFC consists of a set of components written in C++ that wrap necessary functionalities in the development of application for Windows operative systems. It also used a library called csp.dll. After the update, the framework no longer required either MFC or the mentioned library.

The AiClient Framework includes the example bots by David Norman, HoldBot, DumbBot and RandBot, which will be described later. It also includes BaseBot which, as the name suggests, is used as a base for the other three bots. BaseBot is the class that makes the communication with the server and handles every type of message received. This type of messages follow the DAIDE syntax and all the client-server communication is made via TCP/IP, which provides reliable, ordered and error checked delivery. It is a class that can be extended which is the reason it is used as a base for other bots and.

Besides the BaseBot component, the AiClient Framework also provides a way of representing the board itself: the terrestrial, sea and supply center provinces, and the units in Diplomacy . It also provides a way of representing and save the orders that are generated during the game. All these information and representations are handled in the m_map_and_units components.

Regarding the DAIDE's syntax, the Framework also provides a class that tokenizes (creates a token) this syntax, making it easier to generate, parse and handle the messages written, following that syntax.

2.4.6.3 Dipgame

The Dipgame testbed is the newest testbed that allows competition between Diplomacy bots. It was developed by IIIA-CSIC of Barcelona and it is very well documented in phD thesis of Angela Fabregues [14]. Dipgame uses a similar infrastructure to DAIDE with the exception that it provides a parallel server exclusively for negotiation and it is built in Java. Also, to complement this addition, it also includes a negotiation library to allow the use of the L language. The L language is a layered communication standard (Figure 2.4) that defines the way agents can communicate. The communication between clients and server is done by TCP/IP which allows the players to set a game using different computers but the same network.

Dipgame does not support convoy moves. This is because “adding convoys increases the adjudicator’s (module that applies rules) complexity 100% while not providing any benefit from the negotiating point of view”, stated by Angela Fabregues[21].

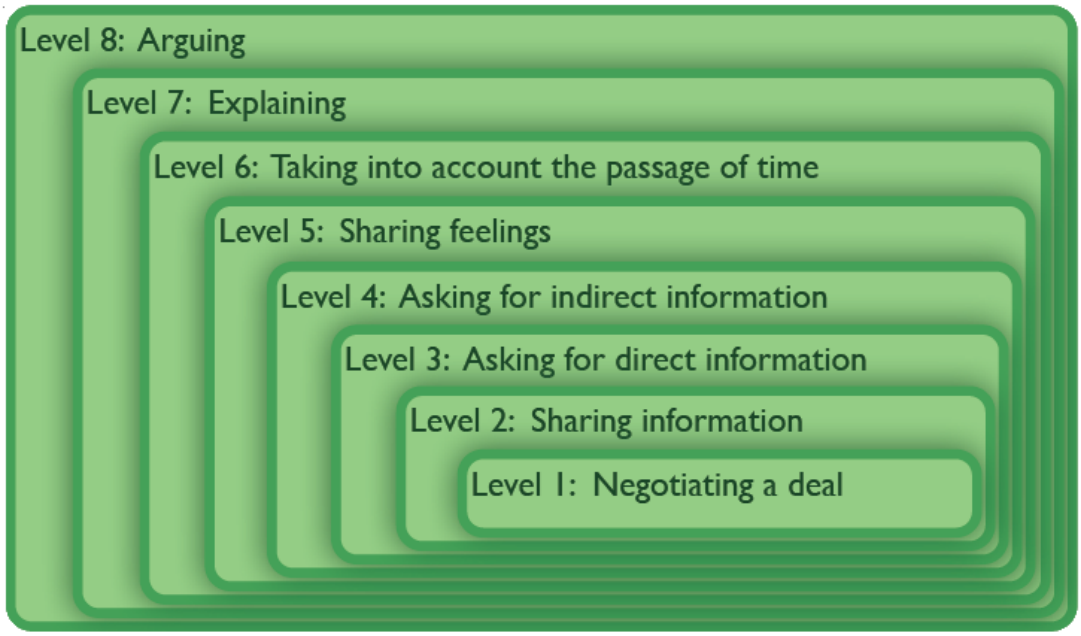

Figure 2.4: Dipgame’s L Language table.

2.4.7 Chosen testbed

In light of the information previously presented, the bot for this dissertation will be developed in the Diplomacy AI Development Environment (DAIDE) using John Newbury’s AIClient framework, updated from David Norman’s previous version. It will also be tested in the same platform. This decision is due to the fact that there already exist bots ready to play press games, while in Dipgame, most of them were difficult to access to. Also, the fact that Dipgame does not support the convoy move was a reason for the outcome decision.

2.5 State-of-the-art

2.5.1 Diplomacy bots

With the creation of the before mentioned testbeds, some bots were created and put into test against each other. Over the years, developers kept creating more and better bots than the ones previously created. Even before the creation of DAIDE, some bots were created with the goal of playing and winning a game of Diplomacy. In this section, a description of some existing bots will be made, starting with the ones created before DAIDE. After those, both DAIDE and Dipgame bots will be described.

2.5.1.1 Israeli Diplomat

The Israeli Diplomat, initially referred to simply as The Diplomat, was, for what is known, the first attempt of creating an AI capable of playing a game of Diplomacy. It was created by Sarit Krauss during her PhD thesis, supervised by Daniel Lehman [22] in the Hebrew University in Israel. Though it was developed almost thirty years ago, the bot is still a source of inspiration to the creation of new bots.

The bot has an architecture with several agents with distinct tasks. This architecture resembles a governmental structure: it has a “Prime Minister”, a “Minister of Defense”, a “Foreign Office”, a “Military Headquarter” and “Intelligence”. The “Prime Minister” module represents the “personality” of the bot. It has a submodule called “Secretary” that contains the bots knowledge of the game, its beliefs, the rules of the game, a message history and information about the other powers. The “Minister of Defense” is in charge of considering different fronts and possible alliances. The “Military Headquarters” takes care of the decision of moves and the ”Intelligence” builds models of others players based on their behavior.

Whenever the Israeli Diplomat begins a negotiation, a new sub-agent is created and is in charge of that specific negotiation. Therefore the bot is capable of multiple negotiations ate the same time (multiple bilateral negotiation). This brings the problem mentioned before of incompatible deals.

2.5.1.2 LA Diplomat

The LA Diplomat was created in the University of California, Los Angeles, by Ari Shapiro, Gil Fuchs and Robert Levinson. The main idea of how the bot works is by learning important strategic aspects of the game through self-play [23]. This is achieved by storing pattern-weights and using temporal difference learning. A pattern is a representation of previous experiences during a game. But instead of saving a full representation of the board, it only saves partial positions of the map or specific moves. The weight of each pattern is used as heuristic evaluation of the state.

2.5.1.3 The Bordeaux Diplomat

Created by Daniel Loeb from Université de Bordeaux, this bot has a similar structure as the Israeli Diplomat. It separates the negotiation module from the search module. This module works with a best first search algorithm. An interesting fact about this bot is that, in order to compute its best orders, it first computes good orders of the other players [24]. To do so, seven parallel processes are launched and each computes the best orders for a power. Then, iteratively, the processes communicate the orders they computed with each other and update their own orders.

The bot also stores the relationship between other players in a matrix. This helps the bot determine which players are more likely to betray it or not.

The three bots described above were not developed for the more recent testbeds. Also it is thought that the source code of the Israeli Diplomat was lost. Luckily, the other two bots (LA Diplomat and Bordeaux Diplomat) have available documentation but the same might not be true for the source code.

Next, an interesting bot will be described. This bot is available for both testbeds mentioned before, Dipgame and DAIDE. It is used both as a benchmark when testing new bots and sometimes as a base from which people start building their own bots.

2.5.1.4 Dumbbot

Dumbbot is a very popular bot in the Diplomacy community. It was created in just two hours, for a challenge, by David Norman[3]. It is a simple bot that does not negotiate but performs well given the circumstances in which it was created. Some of its parameters were chosen by chance, like the values of its heuristics. Alongside two other simple bots, Holdbot and RandBot, Dumbbot is included in the development kits of both DAIDE and Dipgame. Therefore it is very likely to be used as a base bot for beginners as well as a benchmark for testing other bots.

The bot works in two distinct stages being the first stage used to calculate a value for each province and coast. To calculate such values, Dumbbot has the following criteria: if it's its supply center, the size (number of units) of the largest adjacent power; if it's not its supply center, the size of the owning power; if it is not a supply center, zero. The resulting values make it so that Dumbbot tends to make an attack or defend against the strongest player. These values are used to determine not only the moves but also the retreats, builds and disbands. In the second stage, it assigns an order to each of its units trying to move to the adjacent province or coast with the highest value.

2.5.1.5 RandBot

RandBot was also created by David Norman and it is a simple Diplomacy bot that plays randomly: from the available moves each turn, it chooses one randomly.

Next, a description of exclusive bots for DAIDE will be made. The majority of these bots are described in the web page of their creator rather than on a refereed publication. Some of them were actually created as an hobby and not as an academic project.

2.5.1.6 KissMyBot

KMB for short, was created by Jason van Hal. It is considered outdated by its creator given that his newest and best attempt on a Diplomacy AI (Albert) is considered the best Diplomacy bot so far[4]. Nonetheless, it is still a useful bot to use for tests. KMB is able to play a level 10 press game, which means it is capable of establishing peace and alliances with other players. It was later used as a base for the creation of Albert. As KMB is used as basis for Albert and since Albert is considered the best bot so far, not much documentation is available.

[3]Dumbbot Algorithm webpage: http://www.daide.org.uk/index.php?title=DumbBot_Algorithm
[4]Kissmybot webpage: https://sites.google.com/site/diplomacyai/kissmybot

2.5.1.7 Albert

Another Jason van Hal's creation, Albert is considered by the community the best Diplomacy bot so far. It is able to play a level 30 press game, which means it can perform multi-part offers. It uses the previous bot, KissMyBot, as a basis. It is known that Albert deeply thinks about other power's best moves and, iteratively, adjusts the probability of selection of any order set. Also, the creator states that Albert tries "to guess what the other power is going to do"[5]. Albert does have a downside: its good performance comes with a price in term of time consume. The bot takes between five to ten seconds to submit its orders. Of course that this depends on the computer. As the game progresses and Albert gets more units, it gets even slower.

2.5.1.8 BlabBlot

BlabBot was created by John Newbury. It is built over David Norman's Dumbbot. Therefore, we can consider it a Dumbbot capable of simple negotiation[6]. Though it is not clear what press level it is capable of playing, it is stated in [5] that it plays up to level 20 press.

Blabbot's negotiation strategy consists of sending an initial peace proposal to all powers. The powers that accept the proposal are considered friends decreasing the value of that power's provinces. Also, if a power sends a peace proposal to Blabbot, it always accepts the proposal. From that point, Blabbot's actions depend on its Policy. The Policy is an argument passed via command-line and it is optional. If "Joe" is passed, Blabbot remains honest and it will offer a draw if all the remaining powers are friends. Another possible argument is "Mac" which means Blabbot will backstab the remaining players who agreed with peace and it will never offer draws, playing like DumbBot for the rest of the game.

2.5.1.9 Diplominator

Diplominator was created by Adam Webb, Jason Chin, Thomas Wilkins, John Payce and Vincent Dedoyard in 2008. Tactically speaking, Diplominator is quite similar to Dumbbot. The authors state that "As DumbBot uses a similar structure very effectively, we decided to use it as a base to expand upon" [25]. This way, the authors could focus on the negotiation part of the bot. Diplominator does only take one turn to account which means it does not look-ahead in the game. It assigns each province a value based on weather it is a supply center and then uses a blurring algorithm to spread the values to the rest of the board. this allows for a balanced board where provinces between supply centres have a high value and can be considered as a goal by the units. When generating the orders, the bot checks, for each unit, which is the three best moves, choosing one randomly in order to assure its non-determinism.

Regarding negotiation, the bot launches a negotiation thread that processes, in a queue, all the press messages. Once all messages are processed, the thread closes and the orders are submitted. Diplominator uses the peace-to-all strategy in order to know which players can negotiate and which ones are willing to accept peace. It then decreases the values of the provinces owned by them. Besides estab-

[5] ALbert webpage: https://sites.google.com/site/diplomacyai/albert
[6] Blabbot webpage: http://johnnewbury.me.uk/diplomacy/blabbot/

lishing peace, the bot also has the ability to backstab other power when it sees fit. This ability avoids some problems like the one when 2 or 3 powers were allied and kept one blocked in a corner.

Other bots capable of at least level 10 press games are included in DAIDE's clients page[7] and are available for download. Unfortunately, many of them either do not have or could not be found any kind of literature with details about the bot's algorithm and heuristics. Also, there are other bots that do not perform any type of negotiation and only focus on tactical aspects of the game. We have the example of Darkblade, created by João Ribeiro from Universidade de Aveiro [26] and HaAI, created by Fredrik Håård from Blekinge Institute of Technology in 2004 [27] and later improved in 2005 [28].

Following up, some bots capable of negotiation for the Dipgame testbed will be described. Dumbbot is included in the list of bots available but as it is only an adaptation to the L language used in Dipgame, it will not be described again.

2.5.1.10 DipBlue

DipBlue was created by André Ferreira from Universidade do Porto as his Master thesis project [5]. The bot is composed by a flexible and easily extendible architecture. The Advisers are modules that determine the order to be made, while the negotiation is left for DipBlueNegotiator module. Regarding negotiation, DipBlue keeps trust ratio which reflects the relationship with other players. The ratio either increases or decreases depending on the other players actions.

Similarly to Newbury's Blabbot, Dipblue sends a peace proposal to all players. But, as the game progresses, conflicts are unavoidable and so DipBlue breaks the truces with the least beneficial players. To decide who these players are, it takes into account the number of supply centres each player has and his distance in the map to DipBlue.

After peace is established, DipBlue requests alliances with the players who accepted peace against the strongest non-ally player. The target is always updated as the previous one gets weaker.

DipBlue is also capable of suggesting orders to other players regarding their units or accepting requests if the player is an ally and the suggested order has a greater value then the previous order. Nonetheless, the bot may accept the suggested order even the value is lower, taking into account the trust ratio associated with the requesting player.

2.5.1.11 Fabregues' HANA Architecture

During her PhD thesis, Angela Fabregues proposed an agent architecture for resource negotiation problems (refer to [14] for more about resource negotiation problems). The architecture is called Human-Aware Negotiation Architecture, HANA for short, and it was implemented into a nameless agent. The main objective of this bot was to be able to negotiate as humans do, having emotions as major focus. HANA consists of four modules: the interface, the world model, the plan search and the negotiation. The interface is the sensors and actuators. For the world model, it uses a graded Beliefs-Desires-Intentions

[7] DAIDE's Clients: http://www.daide.org.uk/index.php?title=Clients

(BDI). For the plan search, the architecture uses an algorithm that generates complete and joint plans. These joint plans are then used to generate the ranking of negotiations.

2.5.1.12 D-Brane

D-Brane is a bot created by Dave de Jonge as part of his PhD thesis. The name is short for Diplomacy BRAnch & bound NEgotiator. Like many others, the bot has two main components: the strategic and the negotiator components. It uses a NB3 algorithm to search the agreement space and determine which deals to propose [8]. For each deal considered, the strategic component is called in order to calculate the minimum utility value of that deal for each player. Then, the negotiator component uses that value to decide whether to propose said deal or not.

2.5.2 Conclusion

In light of the information previously presented, the bot will be developed in the Diplomacy AI Development Environment (DAIDE) using John Newbury's AIClient framework, updated from David Norman's previous version. It will also be tested in the same platform. This decision is due to the fact that there already exist bots ready to play press games, while in Dipgame, most of them are not operational. Also, the fact that Dipgame does not support the convoy move, made the decision easier.

It was clear that many of the existing bots are composed by modules, and each module has a different functionality. Therefore the architecture planned for this project will also be composed with different modules. Also, it was clear that most negotiating bots perform only low level press (around level 10). Consequently, it is planned that out bot will perform in that level as well, although it is wanted that it performs decently against strongest bot so far, Albert. This architecture will be presented with detail in the next section.

Chapter 3

Tagus Bot's Architecture and Implementation

The following chapter describes Tagus, the bot created in this project. Its architecture, implementation, data structures, heuristics are all described in detail.

The bot was named after the river that runs by Lisbon, Tagus, translated from the Portuguese, Tejo, and the university campus in which it was created.

Tagus is a bot that plays Diplomacy and tries to gain advantage over its opponents by negotiating with them, either to form peace treaties or alliances against other players. Not only that but Tagus also makes use of opening libraries to help it gain some board advantage right at the beginning of the game. It uses some data structures to save information about how trustworthy its opponents are as well as how much tension or friction there is between it and the opponents.

As mentioned before, in order to develop the bot, two different environments were considered, ending up with the Diplomacy AI Development Environment (DAIDE) being chosen. Furthermore, it was used a development kit created by David Norman and updated by John Newbury. This development kit contained all the necessary base material that allowed the created bot to communicate with the server, have a representation of the board and of the plays. This way, it was possible to focus on the implementation of Tagus itself. Nevertheless, it was necessary to understand what the development kit offered and how it worked. Therefore, a description of the development kit will also be made in this chapter.

Figure 2.1 presents a diagram with the architecture of Tagus, displaying its classes, including the ones from the development kit.

3.1 Tagus

Tagus is the core class of the bot. It inherits from the BaseBot class provided by the development kit and it is where the other classes are initiated.

As mentioned before, BaseBot contains the functions used to handle the incoming messages from the server, including press. But, as it is a basic bot, the content of the messages isn't, in fact, treated. It

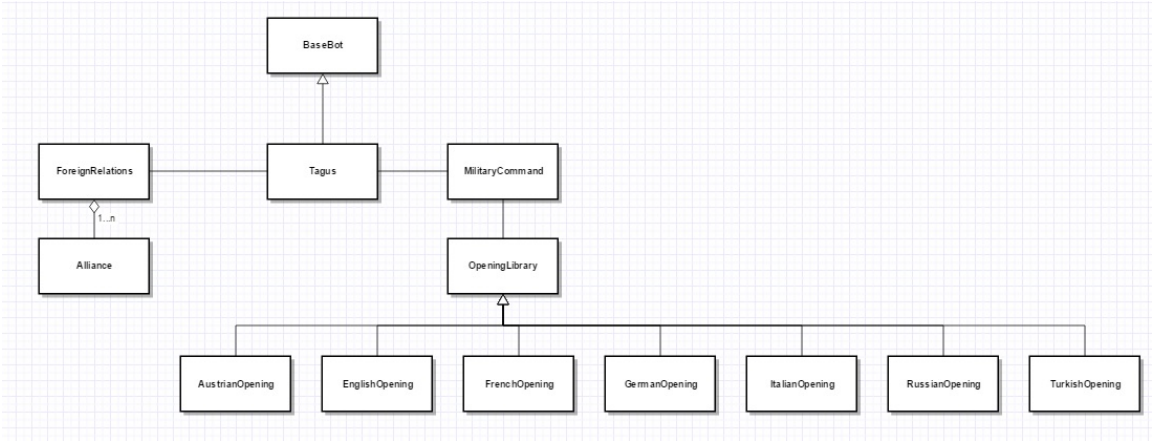

Figure 3.1: Tagus' architecture diagram

is Tagus that handles the information of each message regarding the game itself, i.e, not the connection part. Therefore, one the features of Tagus is to parse the content of the messages following DAIDE's syntax presented in Chapter 2.

Considering all the possibilities of press messages, a parser was built in Tagus class. Besides the FRM type of message, other types with different tags also exist but don't require a parser.

It is this class that also sends proposals to other players (via the server) with the help of the Foreign Relations. As mentioned before, Tagus is a bot for level 10 press games, which means that it is only capable of negotiating peace and alliances.

3.2 Military Command

One of Tagus' component is called the Military Command. This component uses the DumbBot code provided in the Development Kit. Unlike BaseBot, which Tagus directly inherits from, the Military Command does not inherit from DumbBot. It is Tagus' "version" of DumbBot.

As the name suggests, the Military Command is charged with generating orders to each unit every turn. Using DumbBot's algorithm to analyze the board, the Military Command assigns a value to each supply center based upon the strength of the supply center's owner and blurs that value to the surrounding provinces. As such, Tagus, like DumbBot, tends to attack supply centers owned by the largest powers. Then, the orders are generated.

Although DumbBot's code was re-utilized for this project, some changes were made. Changes regarding output writing were made and will be described in Chapter 4.

3.2.1 Province evaluation

The major change was regarding the calculation of the values for each province. Instead of considering only the strength of each supply center's owner, Tagus also considers the relation between itself and the owner of that center. When calculating the defense values, Tagus considers allied provinces as if they were its own. Therefore, Tagus does not discard the possibility of supporting an allied province. As far

as attacking is concerned, when calculating the attack value for a province, Tagus also considers if that province belongs to an ally or to someone who it has a peace treaty with. Nevertheless, Tagus always ponders backstabbing an ally or a peace treaty. Plus, the bot also considers provinces belonging to an "enemy" player, i.e., someone who Tagus is allied against.

To accomplish this behavior, the calculated values are multiplied by a weight, depending on the situation: if it is a province occupied by an ally/peace treaty and the bot is calculating defense values, the value is calculated as if it would be province belonging to it; if it is calculating an attack value and it is and allied province, the value is multiplied by one half (0.5); if the province belongs to an enemy, the attack value for that province is multiplied by one and a half (1.5).

3.2.2 Betrayal

Regarding betrayal, mentioned above, Tagus always checks if it should betray an ally (including peace treaties). The criteria for backstabbing includes the number of supply centers owned by that ally and the trust Tagus has towards that ally. If the ally has ten or more supply centers, the attack value for its provinces will be calculated normally, without multiplying it by 0.5, increasing the possibility of it being attacked. Later it will be explained how Tagus uses trust.

3.2.3 Determinism vs Non-Determinism

To avoid being deterministic and predictable, DumbBot uses randomness when choosing the destination of a unit. After calculating each provinces value, for each unit that DumbBot possesses, it would randomly choose one possible destination amongst the best ones, sometimes not being the best one. For this project, as Tagus would only face other automated players and it was not expected that its moves would be predicted, the random aspect regarding destination selection for each unit was removed. As such, Tagus would only move its units to provinces with the highest value.

3.3 Opening Library

A small addition to Tagus was the use of an opening library depending on the power played. There are seven libraries, one for each power, and they were based on articles written by Richard Hucknall [1]. The strategies are a little more complex than simply trying to establish peace and going for good provinces in the beginning of the game. It also includes long term plans but, as Tagus is a bot for level 10 press, such plans are not possible to elaborate as it requires a higher level of communication.

This component simply consists of a set of powers that Tagus wishes to establish peace with in the beginning of the game and a set of provinces of interest for the beginning of the game as well. In a way, Tagus' first two rounds have a scripted component. The goal is to make Tagus a stronger player in the first few rounds of the game.

[1] http://www.diplomacy-archive.com/resources/strategy/opening.htm

The Opening Library class can be extended to seven different subclasses, one for each power. At the start of the game, after the assignment of the powers to each player, Tagus verifies which power it was assigned with and loads the adequate library. After that, Tagus sends a peace request to the powers contained in the library. Regarding the Military Command, it calculates the values for the provinces considering of interest contained in the library, i.e., multiplying the province value by an associated weight.

3.4 Foreign Relations

The Foreign Relations is the component that gathers the information about alliances and peace treaties between Tagus and the other players. Not only that but it also keeps track of any potential alliance or peace treaty between other players. It is also the Foreign Relations that deals with trust and tension.

Tagus considers that two or more players have an agreement (either peace or alliance in this level of press) if it witnesses a support from either of the players to any other.

3.4.1 Trust Factor

Regarding trust, as the game progresses, the Tagus analyzes other players moves and updates a map with seven entries, each with a value between 5 and -5 associated, starting in 1. That value increases or decreases depending on the other players' moves and what Tagus assumed to be an alliance or peace treaty. If Tagus witnesses an attack from a player to another and they were considered allies of each other, Tagus decreases the trust value associated to the attacker by 2. This value is also increased by 1 anytime Tagus witnesses a cooperative move between two players he previously considered allies of each other. This way, any betrayal would have a greater effect over Tagus' "judgement" along the game. From the information gathered this way, Tagus only proposes peace or alliance to players who have a positive value associated to his/her entry in the map as well as accepting any agreement.

3.4.2 Tension Factor

Another factor that helps in Tagus' way of playing is the tension factor. Like the trust factor, tension is also a value between 5 and -5, starting in 1 as well. This value is used as way to look for peace treaties or alliances against someone and there is a value associated to each other player. As the game progresses, the tension value is decreased by 2 every time a hostile move is made towards Tagus. The value is also increased, by 1, at the beginning of every spring or fall season. When the value is lesser or equal then -1, Tagus will try to arrange an agreement of peace with the hostile power or an alliance against it. Thus, the more hostile moves in a single turn, the quicker Tagus will look for a solution to this tension. As an example, consider the year of 1901: in the beginning of spring season, the initial tension value associated to a certain player is increased (gets better) from 1 to 2. The same player makes a move against Tagus thus decreasing the tension value from 2 to 0. Next, at the beginning of the fall season the value is again increased, going from 0 to 1. But the same player makes another

move against Tagus, making it the second hostile move against it. The value is again decreased, going from 1 to -1. What will happen in winter season, if press is allowed during this phase, is that Tagus will propose peace to that player. In case press is not allowed during summer and winter seasons, explained in Chapter 2, Tagus will wait one more season to propose the above agreements.

The tension factor is modified this way so that Tagus won't look for immediate peace with a player as result of one single attack. It is meant to look for agreements in case of recidivist attacks.

Still regarding the tension factor, Tagus also keeps a 7x7 matrix to record the tension between other players. The values in this matrix are updated the same way as explained before. This is used to check which player would tend to form an alliance against a certain other player.

3.5 Breaking Agreements

Another feature that can be found in Tagus, and as a major aspect of the game, is the ability of breaking an agreement it made and not being bound to it until the end of the game.

In a game of Diplomacy, the criteria for backstabbing can be quite diverse. Tagus' criteria when pondering backstabbing includes the number the number of supply center the other powers have. For instance, if Tagus has an agreement with a power that has a high number of supply centers, in this case 10 or more, a betrayal is most likely to happen. Another criterion involves the trust that Tagus has with that power. If Tagus has low trust in a power with whom it made an agreement with, Tagus rather betray instead of getting betrayed.

There are a few scenarios in which Tagus ponders a betrayal following the criteria mentioned above. One of them is upon receiving a proposal of alliance against an allied power. The other scenario is when the proposal is made by Tagus itself. Another scenario where Tagus ponders to backstab is when selecting the destination for its units. If, as a result of the calculations made by the Military Command, a certain allied province still has a higher attack value than the others, Tagus betrays the owner, breaking any relation with it, and attacks that province.

3.6 Arquitecture at work

Before the game starts, the server sends each player a HLO message that defines which power that player will be playing with. When Tagus receives such message, it loads the corresponding opening library unto the Military Command class (Figure 3.2)

When Tagus receives an ORD message, it updates the trust and tension factor in the Foreign Relations class as Figure 3.3 shows.

Tagus has a dedicated function to deliberate what agreements it will go for. First, it loops through the tension map and sends a peace request to any player whose tension is lesser of equal to -1. In case the trust towards that player is lesser or equal to -1, Tagus looks for an alliance against that player instead (Figure 3.4). This part can be seen as a defensive phase of agreements as Tagus tries to ease any pressure it has.

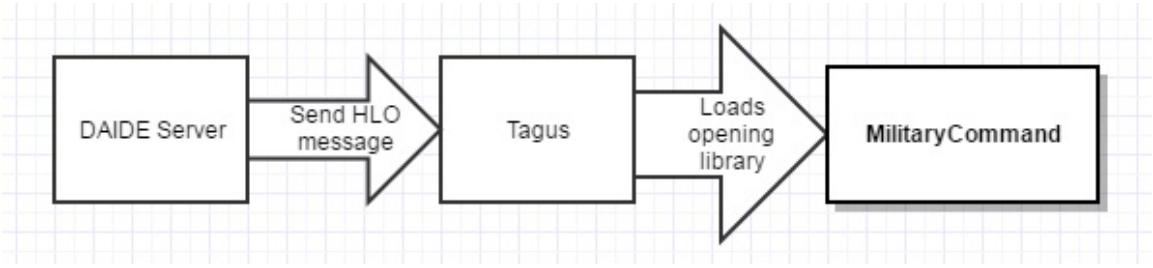

Figure 3.2: Tagus loads the opening library

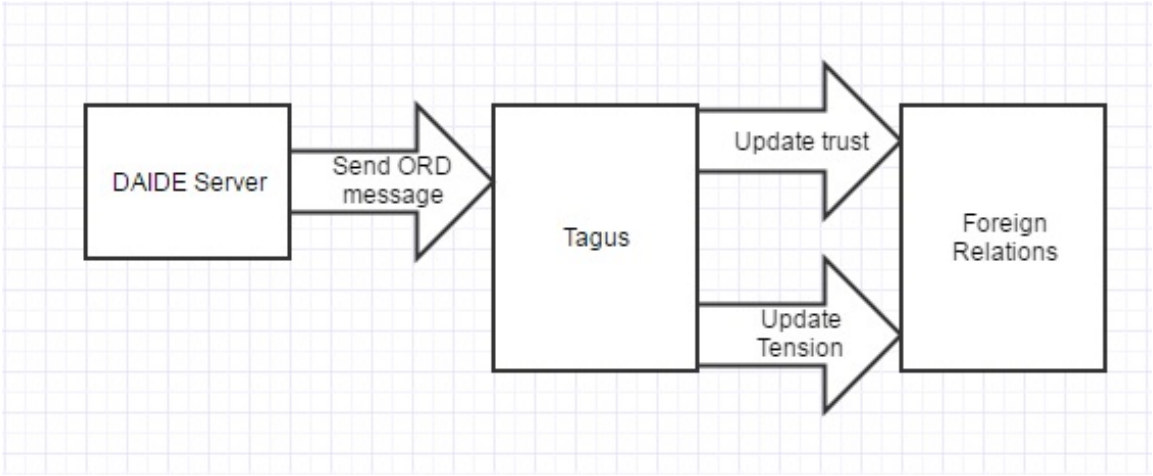

Figure 3.3: Tagus updates the trust and tension factors.

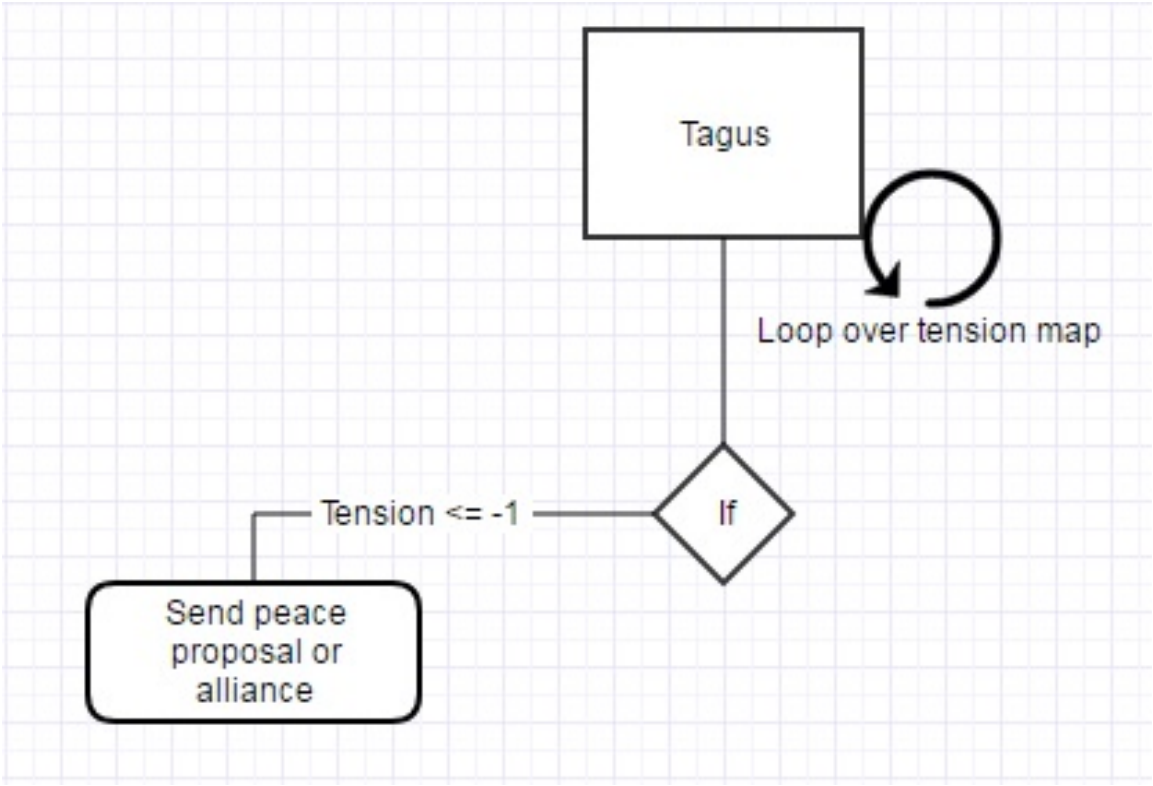

Figure 3.4: Tagus ponders defensive peace or alliances.

After trying a defensive approach, Tagus takes on the offensive. It again enters a loop, only this time to check for players who are too strong. When a player with 10 or more supply centres is found, Tagus tries to form an alliance against him/her (Figure 3.5).

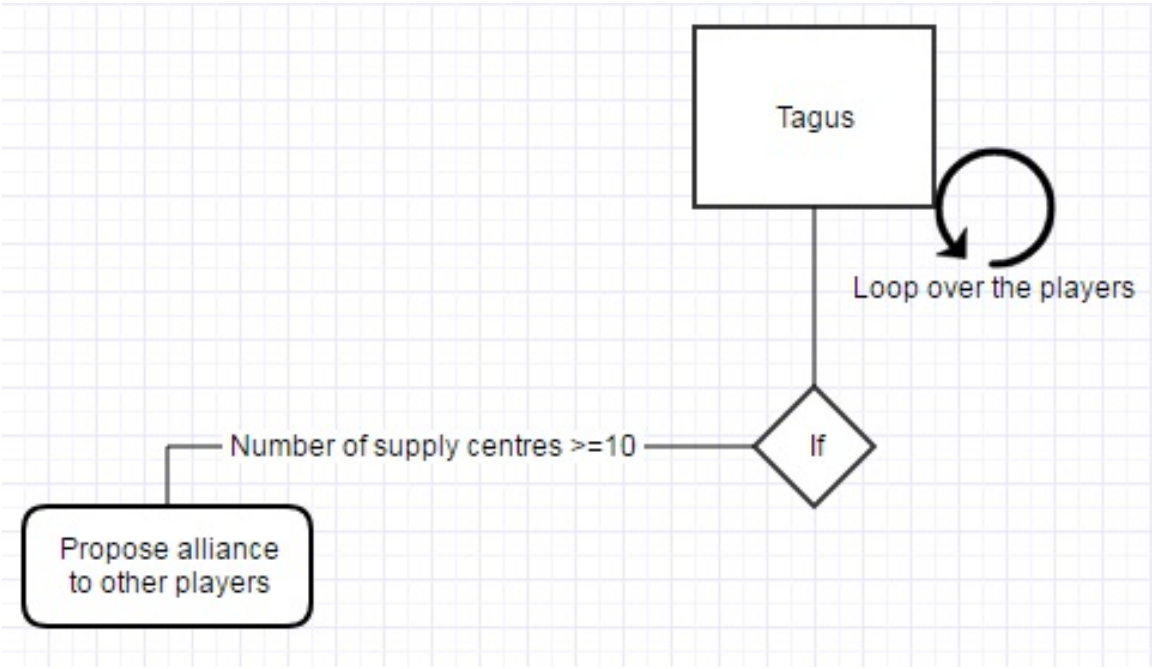

Figure 3.5: Tagus ponders offensive alliances.

When the negotiation phase ends, Tagus starts calculating the value of each province through the Military Command.

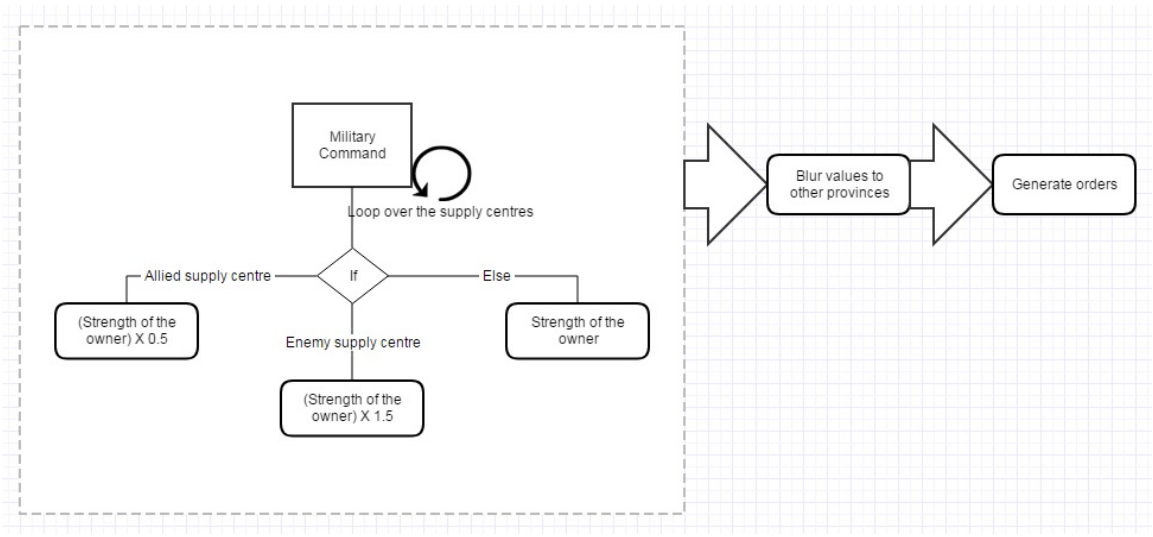

Figure 3.6: Tagus calculates the value for each province.

3.7 Implementation

3.7.1 Installing DAIDE

DAIDE offers several applications to play Diplomacy and test bots.

The essential application is the DAIDE Server from which one can set up a game with the rules he/she desires. Once the game is launched, it waits for the required number of players to join. In a standard game, the application waits for at least 7 players to join. Any further players joining after the first 7 will be observers. A bot can join a game by simply running its executable file. If a human player wants to join, he or she needs the IP address of the machine in which the games was launched.

Another application that is essential for playing but it is not crucial for testing bots is the DAIDE Mapper. This application offers a graphical map which shows the movement of every unit in each turn. Sometimes it is not very useful when testing bots, as bots process the game very fast and it is not possible to follow the movements.

3.7.2 Bot Implementation

3.7.2.1 Tagus and Military Command classes

The first step was a small reorganization of the classes provided by the development kit. As stated earlier in section 2.4.6.2, this kit provided both BaseBot and DumbBot classes. The class of the DumbBot inherited from the BaseBot one and it has methods to handle some important types of messages which are the SCO, NOW and MDF messages. The change made in this situation was the creation of the Tagus class and making it inherit from BaseBot. Then, the code of DumbBot was reused when creating the Military Command. In Tagus class, it was redefined the process_sco, process_mdf and process_now methods which call their versions in the Military Command. Then, the Military Command updates the m_map_and_units according to the information in the messages.

3.7.2.2 Foreign Relations and Alliances

The second step was to create the Alliance class and the Foreign Relations class which are strictly related.

An Alliance object is created every time Tagus successfully establishes an alliance. The class is composed by two lists, one for the allies and another for the enemies. Aside from these lists, it only had two methods, one to add allies and another to add enemies, only used when creating the alliance itself.

The Foreign Relations can be seen as a manager of alliances. It keeps a list of alliances as well as a list of players with whom Tagus made a peace treaty with. The Foreign Relations also keeps the tension matrix (7x7), the tension map and the trust map. Aside from these attributes, the Foreign Alliances has the methods to check weather a player is an ally, and enemy or if he/she is in peace with Tagus. Still regarding its methods, some are used to update the tension matrix and the trust map based on the ORD messages received in the Tagus class.

3.7.2.3 Press

Lastly came the implementation of press exchange abilities.

When Tagus receives a NOW message, it call the doPress method which has three main loops. The first loop is dedicated to the opening libraries, i.e., it cycles through the powers included in the previously loaded library and, depending if the game is still in the first two turns, sends press messages of peace to them.

The second loop can be considered the defensive loop as Tagus cycles through the tension matrix and verifies which players have the most tension with itself. If the player is trustworthy, Tagus sends a preace proposal to him/her, otherwise it loops through the tension matrix again to see which other players enough tension with the target player and tries to form an alliance.

The last main loop can be considered as the offensive one. Tagus checks each other players strength and tries to form alliances against them.

Chapter 4

Experiments

The following chapter describes the methods that were used to experiment and test Tagus' abilities and performance against a series of other known bots. The chapter will also present the results and analysis of the same tests.

4.1 Methodology

To test the developed bot, Tagus, it was first necessary to decide how it would be tested. Therefore, it was created five series of one thousand games each. In each series, Tagus would play against a certain known bot and it was evaluated the negotiation made, the supply center count and the final result of each game.

It was important to evaluate the negotiation that Tagus made, as one of the objectives of this project is to create an automated player that can negotiate with other players, though in this case, it is a simple type of negotiation (level 10 press). Therefore, all negotiation made by Tagus was noted. As each series is composed of 1000 games, negotiation was analyzed only in a few games.

Regarding the number of supply centers throughout the game, Tagus uses an opening library for each power and acts according to said libraries. The goal was to make Tagus a stronger player right at the beginning of the game and try to create a snowball effect during the game. So, this parameter was also noted as number of supply centers per year, per game.

The third parameter that was evaluated was, the result of the game. Either if Tagus won, lost or draw. It is of interest if the bot, after its good (or bad) performance during the game, won said game or not.

All the above parameters are influenced by the power that Tagus is playing with. Tagus may show better performance with certain powers and not such good performance with others. Therefore, it was required a minimum number of games per power. By dividing one thousand games by seven powers, the result would be approximately one hundred and forty-three (143) games per power. Also, all the results were stored in comma separated values files (.csv), for easier treatment.

4.2 Automation

In order to make it possible to run a total of five thousand games, it was necessary to automatize the process of launching the server and the seven bots that were to participate in each series five thousand times. For this purpose, a two options were considered.

MARS the Manage Acquisition of Required Samples is a software used to select, run and monitorize Diplomacy games. Besides MARS, there is also SAGA (Samples of Acquired Game Attributes), which is a database of results of Diplomacy games, and ARENA (Analyses of Results of Experiments Now Available), which contains the latest analyses and discussions of the games stored in SAGA. All these three programs are the main components of DEMO (DAIDE Evolving Model Organizer), a paradigm for automated experimental work. This was the first option considered to run the tests. Unfortunately it was not possible to run this program after following the installation instructions[1].

Scripts A way of automatize a process is to script it. There are many ways of building a script but, for this project, only two were considered. First was the use of batch files. Batch files are script files with a series of commands to be executed by the command-line interpreter. Another option was the use of Powershell scripts, which in concept, are the same as batch files but more modern and are run in a program called Powershell.

The chosen method was the use of Powershell scripts due to its modern and sophisticated attributes which would be more useful than batch files.

After the collection of the game results and the parameters mentioned above, it was also necessary to parse some of the information gathered. This was required due to the size of the files. Some .csv files would get up to 150 000 lines and almost 10 MB. Besides, the file that stored the number of supply centers per power, per game, as well as the winning power and the bot type that played it, had too much information and would be better to separate.

To treat the gather information, four small programs were written in C++. These would read the results file and write the number of wins, losses and draws in one file, the number of games per power in other file, the number of wins per power in another and the supply center count in one other file. This way of separating the information made it easier to analyze the results.

4.3 Test Series

In this section will be described each series of games, reason behind the composition of each of them and the expectations. Each series, as mentioned before, is composed of 1000 games. This number was decided due to the fact that there are a total of seven available powers and in some series there were 3 Tagus playing. Therefore, there are a total of 35 possible combinations of the distribution of powers amongst the 3 Tagus bot. Thus 1000 would give a large enough amount of samples to analyze.

[1] MARS' webpage: http://johnnewbury.me.uk/diplomacy/demo/mars.htm

As some of the games could last indefinitely, it was established a maximum of 10 turns without any board changes. If 10 turns pass and no unit is able to move or any power is able to change its supply center count, the games ends in a draw. The results and respective analysis can be found in Chapter 5.

4.3.1 Series 1: 6 RandBot and 1 Tagus

The first series that was ran was against RandBot. It was a non-negotiation series of 1000 games in which Tagus played against six Randbots. As it is known, RandBot plays randomly but still following the game's rules and legally, i.e, doing legal plays. This was the minimal benchmark.

This series was mainly to test Tagus' playing capabilities. It was necessary to check if any changes made to DumbBots code had negatively influenced the overall bot's performance.

Though this was a non-negotiation series, it was expected that Tagus would get a high percentage of victories due to the fact that it uses DumbBot's algorithm which is superior to RandBot's and due to the fact that Tagus makes use of opening libraries.

4.3.2 Series 2: 6 DumbBot and 1 Tagus

For second series, it was planned that Tagus would play against 6 DumbBots, again, in a non-negotiation game. The expected results would not be as good as in Series 1 because DumbBot has a more complex algorithm of evaluation and plays with non-random criteria.

Though Tagus uses DumbBot's evaluation algorithm, this series has a purpose: to test whether the opening libraries bring any advantage to Tagus or not, i.e, if in the first turns of the game Tagus gathers a good number of supply centers. Otherwise, the series would have no interest as it would be 7 DumbBots playing against each other.

As Tagus is facing 6 DumbBots, in this series, the most relevant factor to evaluate will be the number of supply centers, not discarding the final result. It is expected that Tagus gains advantage in the first turns regarding the number of supply centers but, as the game progresses, the 6 DumbBots will tend to attack Tagus' supply centers. Thus having some difficulties for the rest of the game.

4.3.3 Series 3: 4 DumbBot and 3 Tagus

After the non-negotiation series, the negotiation ones would start. Thus, for the third series, Tagus would face DumbBot again but in different proportions: 4 DumbBots and 3 Tagus. This distribution allows Tagus to negotiate with other bots, in this case, other Tagus bots.

According to the DAIDE's AI Etiquette, found in the DAIDE Overview document[2], trying to identify any other bot of the same type during a game is illegal. In light of that restriction, the 3 Tagus bot behave like they do not know the quality and type of their opponents. As the Tagus bots will not identify each other, they will not recognize the 4 DumbBots as well and, as a result, Tagus will try to negotiate with the DumbBots as it sees fit, though DumbBot will not reply. This restriction applies to the upcoming test series as well.

[2]David Norman's Diplomacy AI Page: http://www.ellought.demon.co.uk/dipai/

For this series, it will be evaluated the negotiation made between the bots in level 10 press games and the result. It is expected that the results will be better than the previous series due to the use of negotiation as well as the increased number of Tagus bot playing. This last factor is what makes the evaluation of how Tagus negotiates relevant. The average number of supply centers will also be taken into account to evaluate the progression of the game, more specifically, the first few turns.

4.3.4 Series 4: 4 BlabBot and 3 Tagus

After testing Tagus against no-press bots, the next series will include only negotiating bots. The fourth testing series comprises 4 BlabBots and 3 Tagus.

In this series, Tagus will be able to negotiate more as all 7 bots have such capacity though BlabBot only negotiates peace.

As mentioned, one of the arguments passed to BlabBot is the -n[Policy] which can be either ”Joe”, ”Mac” or it can be omitted. In this series, the argument was accidentally omitted. This means that ”Optimum ”Tournament” play is then used, the details of which are (deliberately) undefined, possibly varying from game to game”[3]

For this series, negotiation will, again, be a matter of interest as well as the result of each game.

4.3.5 Series 5: 4 Albert and 3 Tagus

So far, all the previous series, except the first one, included bots that either are DumbBot or use it as a base. Series 5 is different in that regard.

In this series, Tagus will play against the best performing Diplomacy bot, Albert, in level 10 press games. The bot distribution will be, again, 4 Alberts and 3 Tagus. Also, in the previous series, there was no time limit in any phase but, for this one, it is necessary to establish time limit for both diplomatic and movement phases as Albert can take 5 to 10 seconds or more if the number of units increases. Each one of this phases will have 3 seconds each.

As previously mentioned, it is not known how Albert actually works. It is only known that it may take time for it to make its moves and that it somehow predicts other players. Thus, the expectations for this series are quite open. Tagus' performance will depend on how Albert treats alliances and peace.

This series is the only one where Tagus will be able to form alliances with any other player which can be positive to its performance.

Like the previous series, it will be evaluated the negotiation made by Tagus as well as the result of the games. Regarding this last parameter, and as a guess, Tagus will struggle against Albert.

[3]John Newbury's BlabBot description: http://johnnewbury.me.uk/diplomacy/blabbot/index.htm

Chapter 5

Results

The following chapter is dedicated to the analysis of the series previously described. The results will be shown following the criteria presented in the series' description. The graphics shown, regarding the average number of supply centers, may show an inconsistent line in later turns (120 and later) because the number of games with such high quantity of turns is low. Thus, the average may result of a low number games with a large difference of supply centers. Also, these numbers were counted at the end of each year, which explains why in the first years shows a higher number than the starting one for each player.

Each section will describe the results of each series. Two graphics are presented in each section with the first showing the average of supply centers and the second showing the number of games (Y axis) that took a certain amount of turns (X axis).

5.1 Series 1: 6 RandBot and 1 Tagus

The first series had the expected good results. Tagus won 987 games out of 1000, a percentage of 98,7, and there were no draws. Table 4.1 shows the number of games per power as well as the percentage of wins.

	Tagus		
Power	# of Games	# of Wins	% of Wins
Austria	147	146	99,32
England	145	141	97,24
France	131	129	98,47
Germany	138	136	98,55
Italy	131	130	97,24
Russia	159	159	100
Turkey	149	146	97,98

Table 5.1: Results of series 1 with: number of games per power, number of wins per power, percentage of wins of games played per power .

Regarding the number of supply centers, there was a significant increase in the first 8 turns of the game, having some games ending between the 8th and 10th turn. Games that took longer than 10 turn

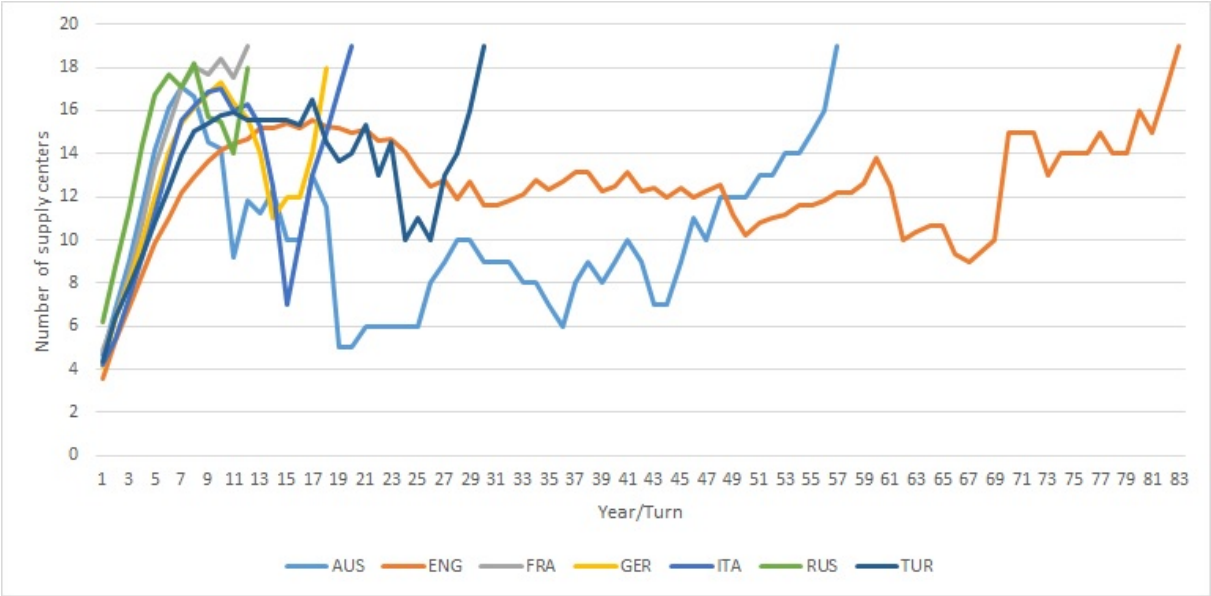

Figure 5.1: Series 1: average supply center count per year per power.

had a slower progression in terms of supply centers. Figure 4.1 shows the average supply center count per year, per power. It is shown that Austria and England had the longer games but, in conjunction with Figure 4.2, it is noticeable that the number of games that exceeded the 25 turn mark is small, having only one game reaching 83 turns with England. Also, looking at Figure 4.2, it can be confirmed that most of the games did not last more than 5-11 turns, as seen by the abrupt fall around that interval.

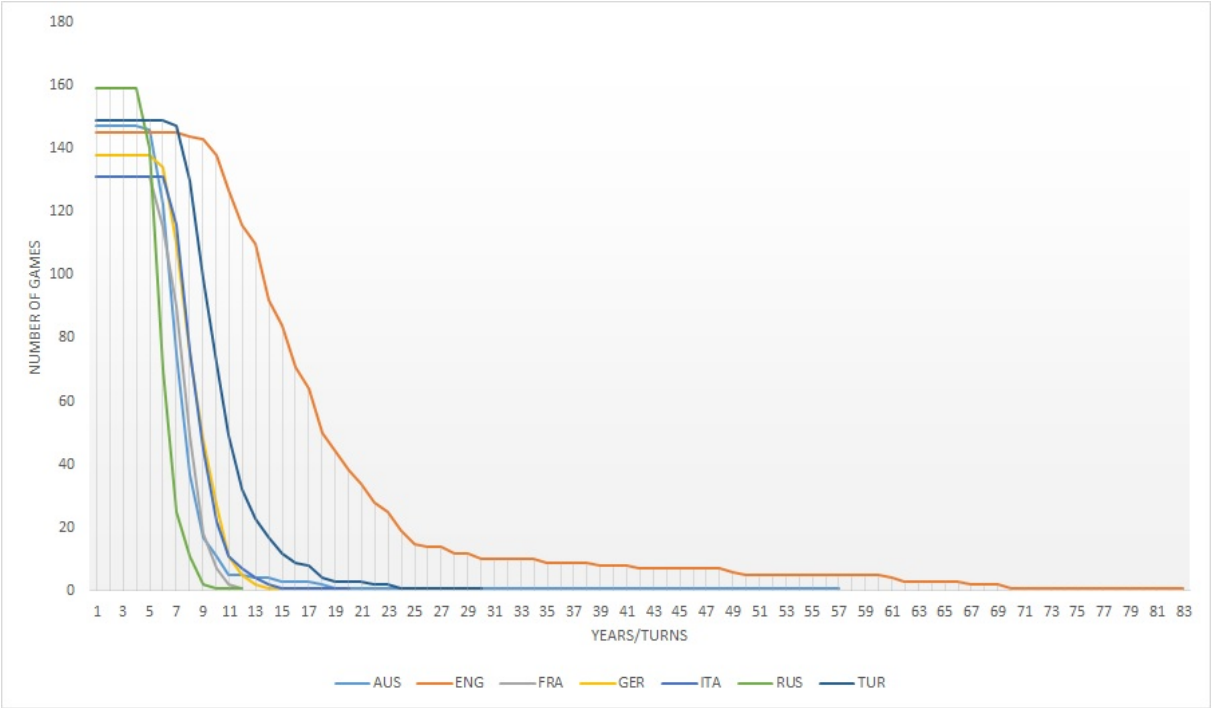

Figure 5.2: Series 1: number of turns per game.

5.2 Series 2: 6 DumbBot and 1 Tagus

The second test series consisted of 1 Tagus playing and 6 DumbBots. Like the previous one, it is composed of 1000 games of level 0 press as only one of the participating bots (Tagus) is capable of negotiating.

In this series, Tagus, as expected, showed a poor performance regarding win percentages. Out of 1000 games, Tagus was able to win only 59 games, with the distribution shown in Table 4.2.

	Tagus			DumbBot		
Power	# of Games	# of Wins	% of Wins	# of Games	# of Wins	% of Wins
Austria	133	0	0	866	74	8,54
England	146	21	14,38	852	299	35,1
France	153	12	13,27	845	132	15,62
Germany	127	0	0	885	4	0,45
Italy	141	5	11,97	857	111	12,95
Russia	150	13	12,28	848	44	5,19
Turkey	150	8	7,45	835	107	12,81

Table 5.2: Results of series 2 and comparison between the bots.

By not being able to negotiate with other players, Tagus was tied to its opening strategies and to its ”DumbBot behavior” in order to win. Figure 4.3 shows Tagus' supply center average. It is possible to verify that Tagus best performed as England in this series. It is also possible to see that, in one hand, with England, Turkey and France, there is an increase in the average number of supply centers followed by a period of stabilization or growth. On the other hand there is a quick fall while playing with other powers.

The poor results of this series are also related to the ”nature” of DumbBot. As stated in section 1.2.5, its algorithm focuses on moving to supply centers owned by the stronger players, i.e, players with higher number of supply centers. Which means that, with the initial increase of supply centers, Tagus becomes somewhat prioritized in being attacked.

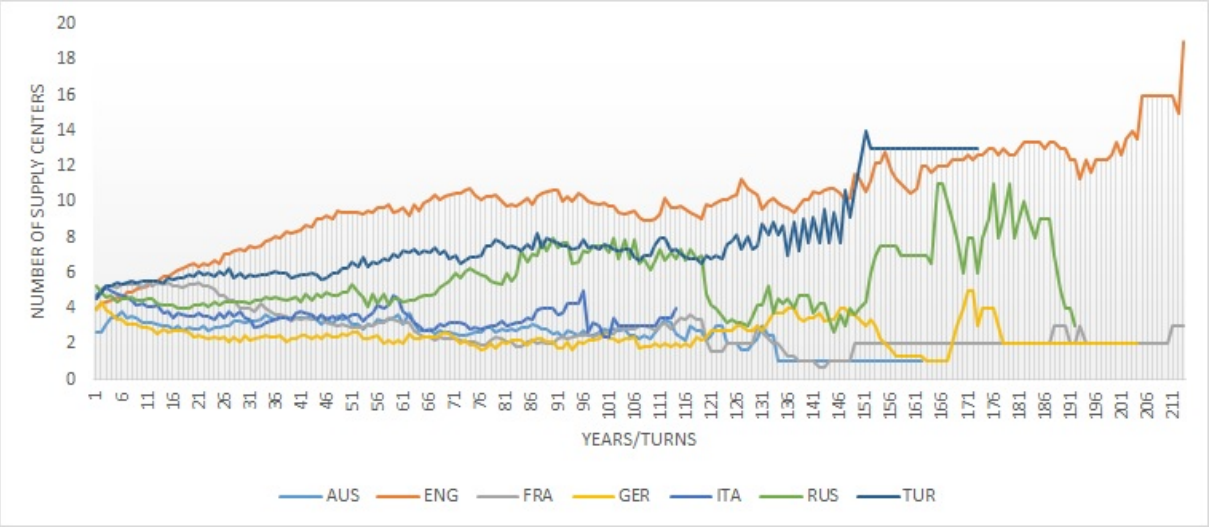

Figure 5.3: Series 2: average supply center count per year per power.

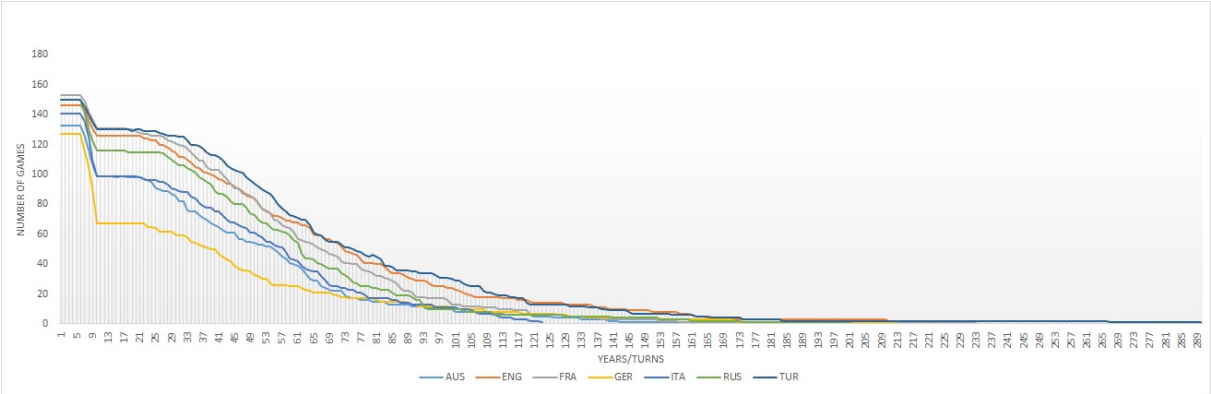

Figure 5.4: Series 2: number of turns per game.

5.3 Series 3: 4 DumbBot and 3 Tagus

In the third series, Tagus played, again, against DumbBot but this time in a 3x4 distribution, i.e, 3 Tagus and 4 DumbBots. Note that this does not mean 3 Tagus vs 4 DumbBots. Each Tagus can and will play against bots of the same time. Also, as there are 3 Tagus bot playing, the number of games played with each power increases. The results did not reach the minimal number of wins, which would be 3/7 of 1000, around 429 wins. Instead Tagus accomplished only 340 wins in this series and 133 draws. Although it was not the minimal number of wins, 340 is not a value too far from the such minimal, considering the total number of games.

	Tagus			DumbBot		
Power	# of Games	# of Wins	% of Wins	# of Games	# of Wins	% of Wins
Austria	450	24	5,33	551	50	9,07
England	472	113	23,94	529	168	31,76
France	437	58	13,27	563	93	16,52
Germany	391	13	3,32	610	5	0,81
Italy	426	51	11,97	575	83	14,43
Russia	399	49	12,28	602	33	5,48
Turkey	429	32	7,45	572	67	11,71

Table 5.3: Results of series 3 and comparison between the bots.

Due to the inconsistency found in the results regarding the number of supply centers during the games and the large number of games, the average of that same factor resulted in a overall low value. Tagus had games in which it dominated, with a good curve of supply centers acquired but on the other, there were games in which its performance was quite poor.

By looking at the graphic in Figure 4.3 and table 4.2, it is conclusive that Tagus had a better performance when playing as England. On the other hand, while playing as Russia, its loss of supply centers is right at the beginning of the game. Also, while playing as Germany, Tagus also shows some struggle by loosing supply centers and hardly recovering from that loss. This shows that the opening library for these countries did not work very well. As far as the other are concerned, Tagus shows a small growth and stabilize until later in the game.

While playing as France and Turkey, the data shows that in long games, the average number of supply

centers increases. This fact happens because, in later turns, there are less players alive, which means that is easier to ”focus” on players who are getting too strong and form alliances more efficiently. This happened more often when the surviving players were of the Tagus type. Also, graphically speaking, the increase shown is also due to the fact that the number of games with a high turn extension is lower, thus influencing the average calculation.

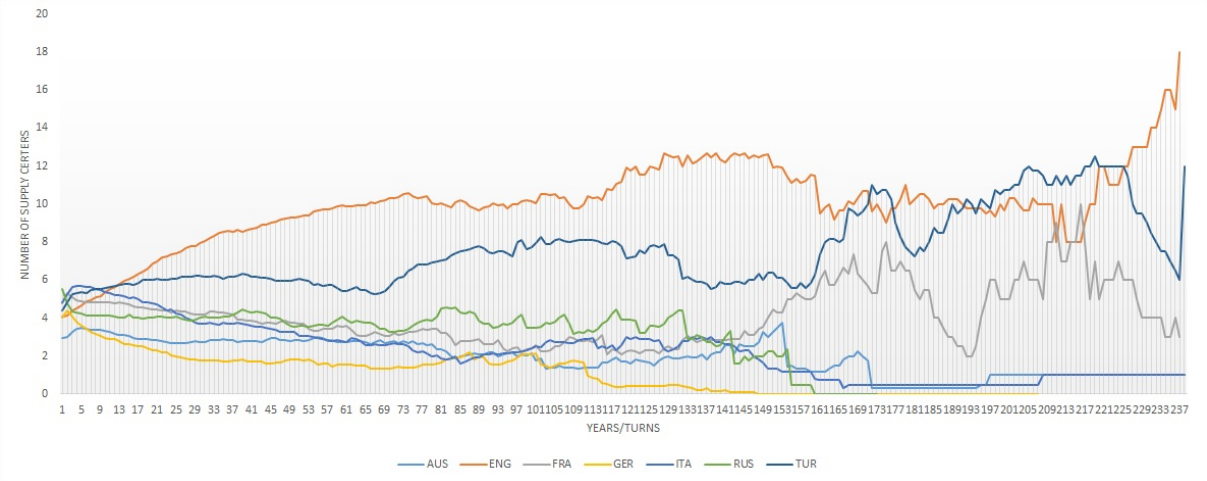

Figure 5.5: Series 3: average supply center count per year per power.

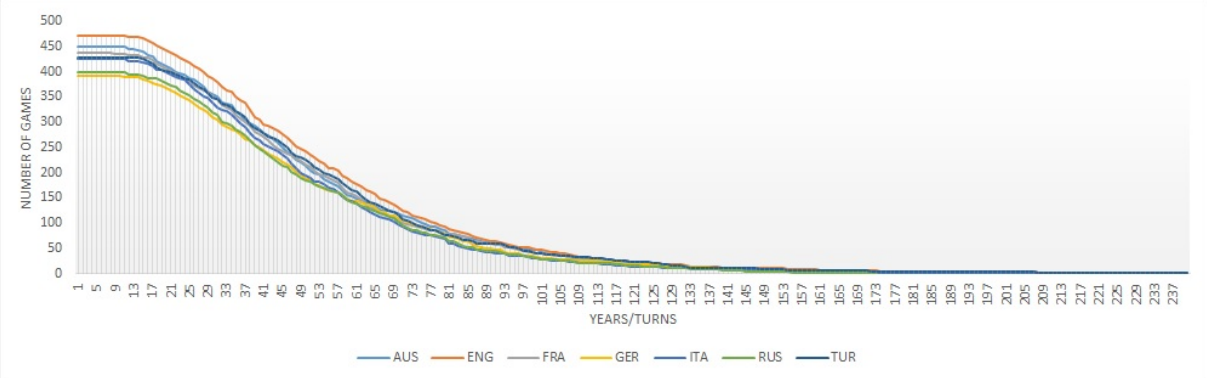

Figure 5.6: Series 3: number of turns per game.

5.4 Series 4: 4 BlabBot and 3 Tagus

The fourth series of test to be run was the first to have seven negotiating bots. The games were played by 4 BlabBots and, again, 3 Tagus. It is know that BlabBot's negotiating capacities are limited to only peace and draw proposals. Nevertheless, it was enough to run a series where Tagus could negotiate with any of the players, contrary to the previous. Tagus was able to get a total of 361 wins, 79 draws and 560 losses. Table 4.4 shows the number of games per power and the corresponding winning percentage.

As mentioned before, BlabBot's ”policy” parameter was omitted, which resulted in an unknown behavior. After further analysis of the orders of each player and the press exchange between them, it was possible to conclude that BlabBots were using a policy resembling ”Mac” since there was not any draw

	Tagus			BlabBot		
Power	# of Games	# of Wins	% of Wins	# of Games	# of Wins	% of Wins
Austria	452	70	15,48	548	28	5,11
England	419	6	1,43	581	32	5,51
France	419	113	26,97	582	137	23,54
Germany	450	39	8,66	550	51	9,27
Italy	408	20	4,90	592	97	16,39
Russia	431	66	15,31	569	123	21,62
Turkey	422	47	11,14	578	46	7,96

Table 5.4: Results of series 4 and comparison between the bots.

proposal and BlabBot would attack players who were considered ”friends”. The omission could have negatively influenced the series if the policy resultant of it was not desired.

The graphic from Figure 4.5 shows the average progression regarding supply center for this series. Though most victories were achieved while playing as France, Turkey shows a better average curve, having a positive growth and stabilizing for most games. It is clear that Tagus struggled while playing as Italy and, oddly, its performance as England highly dropped compared to the previous series.

Although the number of victories was not too bad, the graphic shows inconsistency in Tagus performance during this series, having an overall low average. This mean that Tagus performed very well in some games and very poorly in others. Also, the abrupt fall, rise or the constancy in some powers' curve, for example England in turn 81 or Turkey in turn 127, is caused by the small number of games in which the number of turns was higher.

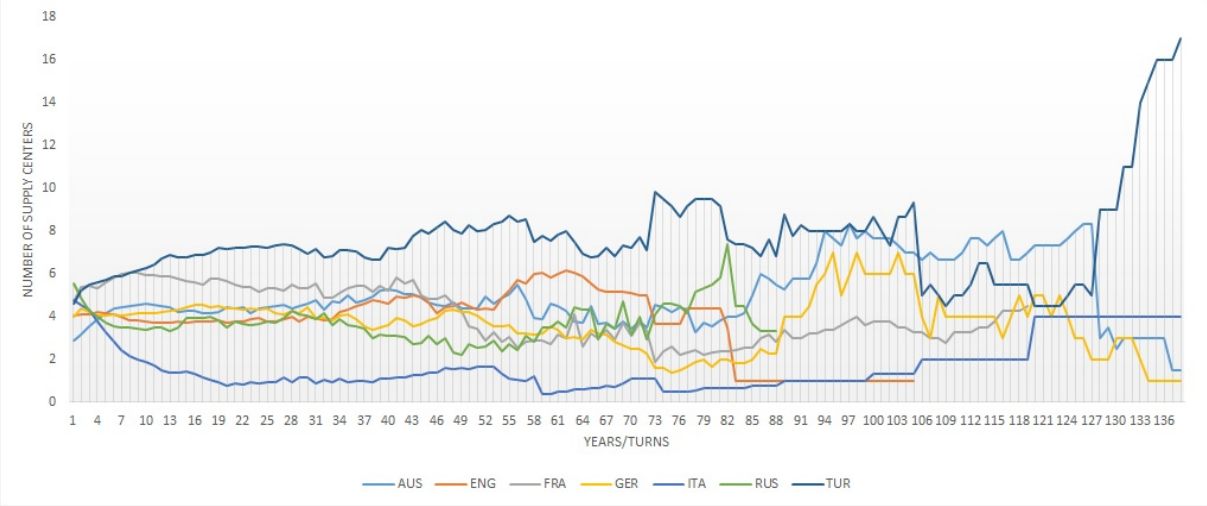

Figure 5.7: Series 4: average supply center count per year per power.

Something that was possible to identify in this series was the use of the betrayal feature. At some point of a game, Tagus, playing as France, achieved 8 supply centers in first 8 turns until it started having recidivist confronts with Italy, played by a BlabBot. The tension grew and Tagus asked for peace to Italy in 1906. The agreement was accepted. Later, in 1909, Italy reached the amount of 10 supply centers. In the same year, Tagus looked for an alliance with Austria and Turkey (separately) against Italy. The alliance was accepted by Austria, played by another Tagus and was answered with ”BWX” by Turkey, played by another BlabBot. From that point, Italy's strength started falling and ”french” Tagus was able

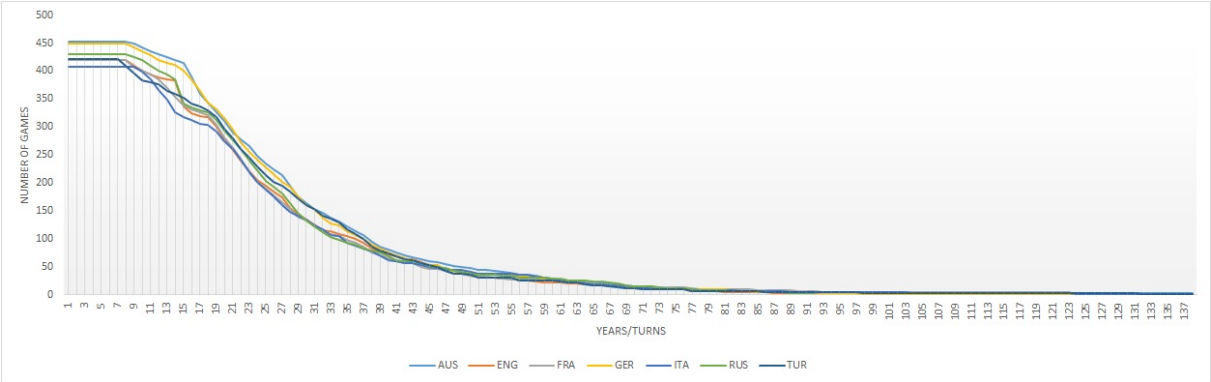

Figure 5.8: Series 4: number of turns per game.

to grow stronger.

5.5 Series 5: 4 Albert and 3 Tagus

The last series of tests was the one with the most curious results. It was composed of 1000 games with 3 Tagus and 4 Alberts. Tagus achieved 236 victories which, again, is below the minimal but, considering the high number of games played, Tagus only lost 316. This means the series had a total of 448 draws, making it the one with the highest number. These results were surprising considering the ones from series 4, where Tagus played against a supposedly weaker bot. Table 4.5 shows the number of games played with each power as well as the number of wins and percentages.

	Tagus			Albert		
Power	# of Games	# of Wins	% of Wins	# of Games	# of Wins	% of Wins
Austria	442	22	4,98	582	46	7,9
England	389	7	1,80	537	58	10,8
France	434	72	16,59	566	61	10,78
Germany	459	21	4,58	563	35	6,22
Italy	408	25	6,18	599	42	7,01
Russia	431	54	12,53	574	40	6,97
Turkey	423	35	8,27	579	34	5,87

Table 5.5: Results of series 5 and comparison between the bots.

After the analysis of a few games, it was checked that most of the proposals sent to a power played by Albert were never answered. The only situations which a Tagus could make an agreement with a Albert was when the proposal was made by Albert. A reason behind this event is the time limit established in each movement and negotiation phase. Each phase had a time limit of 3 seconds which probably were not enough for Albert to process the proposals. This fact highly influenced Tagus' performance as it could not establish agreements whenever it saw necessary. The limit turned out to be an essential factor in order to make this series doable. Even with it, the series took around 5 days to complete.

The time limit for each phase may also be the reason for the high number of draws. The author of the bot, Jason Van Hal states that Albert takes "(...)between 5-10 seconds(...)" to submit its others and that "(...)it does slow down when powers get to a greater number of units" which is "(...) not ideal for

bot-bot tournaments"[1]. As explained before, the server would declare a draw if, after 10 turns, there was no change to the supply center ownership. In light of all this information, it is possible that Albert could not finish the games while still having the capacity of not allowing any other bot (Tagus or Albert) to win.

The graphic of Figure 4.6 shows that Tagus had an overall good performance while playing as Turkey, having a good progression in the number of supply centers. It also shows a decent progression with England though the results were not as good. As France, which was the power that Tagus obtained more victories in this series, the progression was not as good. Like in the previous series, the last 20-30 turns show odd curves which match the low number of games that actually reached such high amounts of turns.

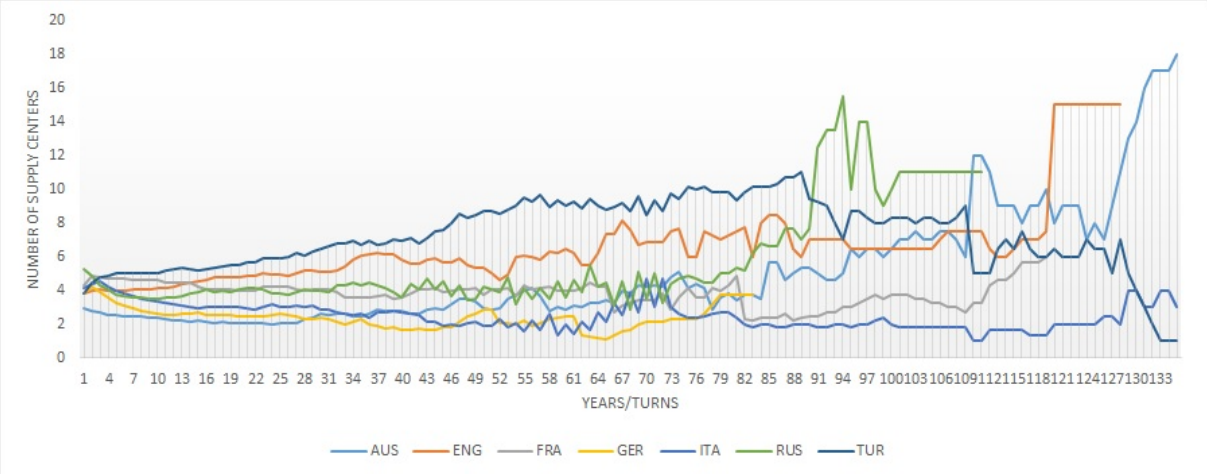

Figure 5.9: Series 5: average supply center count per year per power.

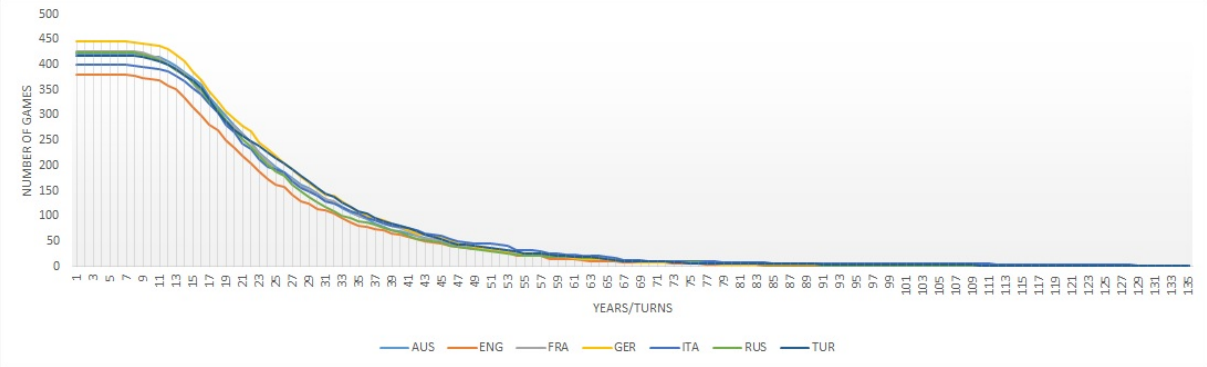

Figure 5.10: Series 5: number of turns per game.

5.6 Summary

The five series of tests had the purpose of documenting Tagus' performance while playing several games of Diplomacy against different opponents. Each series had a different goal and it was possible, through

[1] Albert's description webpage: https://sites.google.com/site/diplomacyai/albert

the analysis of the results, to conclude that each series was successful, although the outcome of the games was not as good as wanted.

It was clear that Tagus benefited more from the opening libraries while playing as France, England, Turkey and, in some series, Austria. As far as the other power are concerned, Tagus did not benefit as much.

Regarding negotiation, which was the most important aspect of these tests, the results are satisfactory. Tagus tries to negotiate whenever it is possible though sometimes it showed a exhaustive behavior. When Tagus sends a negotiation and gets a negative reply or no reply at all, it keeps trying the same deal in upcoming turns.

Overall the results were satisfactory.

Chapter 6

Conclusions

This chapter describes the resulting conclusions of the work previously presented and how the objectives were reached. Also some ideas for future work will be presented and how Tagus can be improved.

6.1 Achievements

The project described in this dissertation was, overall, successful. It meet the main goal which was to build an artificial intelligence capable of playing Diplomacy. Tagus is a bot fully capable of playing level 10 press games following DAIDE's rules and the fact that it plays in such level means it can perform simple negotiation.

The main goal was achievable by successfully completing the three subgoals presented in Chapter 1.

6.1.1 Subgoal #1

The first subgoal, to "create a functional automated player for board game Diplomacy", was successfully achieved with the help of David Norman's development kit. It provided the basic but essential material to build a simple Diplomacy bot and start the project. Tagus' main class was created, extending the provided BaseBot class and, by using DumbBot, Tagus become a playable bot though with nothing new. This was not an issue regarding the subgoal as, overall, the project focused more in its negotiation abilities.

6.1.2 Subgoal #2

With the first subgoal complete, i.e, with a fully functional bot, it was possible to start working on its enhancements. The first one, which was the second subgoal of this project, consisted on the implementation of the opening libraries. These turned out be a very flexible component of the bot as more can easily be created, as well as more complex ones.

The results show that some opening libraries were more effective than others. Taking France as an example, the library helped Tagus grow stronger in the first turns of the game by focusing on key supply

centers. With some other powers, for example Italy, the library was not very helpful mainly because the supply centers it would focus on were contested most of the times.

Overall, this subgoal was successful as in some cases Tagus' was able to take advantage of the libraries. Nevertheless, there is room for improvements, described in the section "Future Work".

6.1.3 Subgoal #3

The third subgoal of this project, and the one in which more effort was spent to complete, focused on implementing negotiation abilities to the bot. This subgoal would make the difference between having a bot solely capable of playing level 0 press games or higher, in this case, level 10. Presently, Tagus is a successful negotiating bot for Diplomacy, being able to play up until level 10 press games. It negotiates peace and alliances with other players as either an effort to protect itself or in an offensive way.

Still regarding this subgoal, Tagus successfully ponders betrayals and executes them when it sees fit. This was witnessed and documented in Chapter 5.

6.2 Theoretical conclusions

6.2.1 Diplomacy and Automated Negotiation

Diplomacy is a game where players negotiate with the purpose of gaining something positive out of it. A simple example can be the negotiation of alliances. When forming alliances, the players look to maximize their strength against others. This strength can be regarded as an utility mentioned in section 2.1.1. Another example of utility, although a little more complex in level 10 press games, is the number of supply centers. In higher levels, players can propose the yield of a certain supply center up until a certain year in exchange of something else, e.g., information or the support in a future movement.

The number of possible agreements that players can make during a game of Diplomacy is large, possibly infinite, as players can negotiate whatever they want in any way they want. Nevertheless, if the complexity of possible agreements is lowered, for example, in level 10 press games, the number gets smaller and finite, and it is possible to build a set of agreements for each turn.

Finally, in Diplomacy players are not forced to keep the agreements they make. They can betray the counterpart whenever they want and as they see fit. Contrary to this, automated negotiation considers that any agreement made by the agents must be kept until its end (section 2.1.1). This fact would exclude Diplomacy from the field of automated negotiation. Nevertheless, in light of what was said above, instead of excluding Diplomacy from this field, it could be included as particular case of automated negotiation.

6.2.2 Tagus' Negotiation Protocol

In Diplomacy, players can negotiate with more than one other player. If we take for example a three way negotiation between France, England and Germany, Diplomacy would be included in the multilateral group regarding its negotiation protocol, as stated in section 2.1.3. In this dissertation though, Tagus

was build as if it would be playing in a multiple bilateral protocol environment, i.e., instead of sending alliance proposal for more than one other player, it send only to one at a time. This decision was made to assure that the reply would depend on the decision of one player and not more. As an example, if Tagus, as Russia, sent an alliance proposal to Turkey, Austria and Italy against Germany, France and England, and one of them would reject the proposal or simply would not answer, in the DAIDE environment, this would result in a rejection and no alliance would be made. Instead, Tagus sends three proposal.

It is possible to conclude that, even though Tagus is included in a multilateral negotiation protocol environment, its own protocol is a multiple bilateral one.

6.3 Future Work

Although Tagus turned out be a satisfactory result of the work of this project, there's still plenty of room to improve. In this section, several potential improvements to Tagus will be proposed as future work.

6.3.1 Press Level

As showed in section 2.4.6, DAIDE has 15 possible levels of press. In the current state, Tagus is capable of playing only up to the second level, being the fourth the highest level achieved by Albert.

Initially, Tagus was projected to reached the third level of press (Level 20, Order Proposals) but, as the knowledge about the testbed was short and some effort was spent on understanding it, the goal was lowered to the second level. Therefore, one proposition for future work is to enhance Tagus' capabilities to reach higher levels of press. This improvement will allow Tagus to play more complex games as well as improving other components described below.

6.3.2 Opening Libraries

After analyzing the results of the test series, it was clear that, while playing as certain powers, Tagus did not benefit much from the opening libraries.

The source from which the libraries were based upon included negotiation that would reach level 40 (Sharing out supply centers) press games, as it would require the yield of some supply centers in a certain year in exchange of something else. As mentioned before, Tagus is a level 10 press bot and , presently, is far from level 40.

Another problem with the libraries was that some would focus on non supply center provinces. This was problematic because of how Tagus evaluates the map. Using an adaption of DumbBot's evaluation algorithm, Tagus first gives a numeric value, which can be easily modified, to all supply centers and then blurs that value to the surrounding provinces. This fact was mistakenly forgotten, resulting on the weights of non supply center provinces, included in the libraries, being ignored.

For future work, it is proposed a review on the opening libraries considering the problems mentioned above.

6.3.3 Negotiation

One of the main features of Tagus was the ability to negotiate with other players regarding alliances and peace treaties. As said before, this was satisfactorily achieved but, right now, it follows a very simple heuristic.

Tagus will only look for alliances when some other player is getting too strong or when it is constantly attacked by a particular one. It would be interesting to include other factors to Tagus' negotiation algorithm, which would be useful when the press level increases.

Still regarding negotiation, Tagus way of negotiation could and should be tuned a little more. Presently Tagus does not keep a record of previous agreements it made. Thus, sometimes it breaks an agreement with another player and, if that same player still fulfills the requirements to negotiate with, Tagus will try to negotiate with him again in the next turn. Also, after sending a proposal to a player, if the player does not reply of replies negatively, Tagus keeps trying to reach an agreement with him. That is why it was said before that Tagus tries to negotiate in an exhaustive way.

Lastly, a good improvement to Tagus would be the use of the TME type of message. These messages allow the players to know how far they are from the time limit in each turn. This would allow Tagus to send a proposal, wait for the reply and proceed with other proposals or the orders of the turns. Presently, Tagus sends the proposals without waiting for the reply of previous ones and sometimes proceeds to build the orders without all the replies, resulting in a not optimal turn.

6.3.4 New test series

In this project, Tagus was tested in 5 different series, each with different purposes. It played against four different bots but DAIDE offers many more which could be used for future test series. An interesting series would be 1 Tagus and 7 other different bots.

Bibliography

[1] Ian Millington and John Funge. *Artificial Intelligence for Games, Second Edition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2009.

[2] Rèmi Munos Olivier Teytaud Sylvain Gelly, Yizao Wang. Modification of UCT with Patterns in Monte-Carlo Go. Technical report, INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE, 11 2006.

[3] Chris Duckett. Google alphago ai clean sweeps european go champion. *Innovation*, 2016. Retrieved from http://www.zdnet.com/article/google-alphago-ai-clean-sweeps-european-go-champion/.

[4] Allan Calhamer. The invention of diplomacy. diplomacy-archive.com, 1974. Retrieved from http://www.diplomacy-archive.com/resources/calhamer/invention.htm.

[5] André Ferreira. Dipblue: a diplomacy agent with strategic and trust reasoning. Master's thesis, Faculdade de Engenharia da Universidade do Porto, 2014.

[6] Fernando Lopes, Michael Wooldridge, and Augusto Novais. Negotiation among autonomous computational agents: principles, analysis and challenges. *Artificial Intelligence Review*, (29):1–44, 2008. Retrieved from http://www.cs.ox.ac.uk/people/michael.wooldridge/pubs/ai-review2008.pdf.

[7] Roy Lewicki, Bruce Barry, David Saunders, John Minton, Bruce Barry, and John Minton. *Negotiation*. McGraw-Hill/Irwin, 2002.

[8] Dave de Jonge. *Negotiation over Large Agreement Spaces*. PhD thesis, Universitat Autónoma de Barcelona, 2015.

[9] Nicholas Jennings, Peyman Faratin, Alession Lomuscio, Simon Parsons, Michael Wooldridge, and Carles Sierra. Automated negotiation: Prospects, methods and challenges. *Group Decision and Negotiation*, (10):199–215, 2001. Retrieved from http://www.cgwmuseum.org/galleries/issues/cgw_75.pdf.

[10] Warren Raisch. *The eMarketplace: Strategies for Success in B2B eCommerce*. McGraw-Hill, Inc., 2002.

[11] Orlando Pinho Jr., Geber Ramalho, Gustavo Paula, and Patrícia Tedesco. *Advances in Artificial Intelligence*. Springer, 2004.

[12] Wizards of the Coast. *Diplomacy rules book*, 2014.

[13] Alan Ritchie. Diplomacy - a.i. Master's thesis, University of Glasgow, 2003.

[14] Angela Fabregues. *Facing the Challenge of Automated Negotiation with Humans*. PhD thesis, Universitat Autónoma de Barcelona, 2012.

[15] John Collins, Maksim Tsvetovat, Bamshad Mobasher, and Maria Gini. Magnet: A multi-agent contracting system for plan execution. In *Proceedings of SIGMAN 98*, 1998.

[16] Michael Wellman and Peter R. Wurman. A trading agent competition for the research community. In *In IJCAI-99 Workshop on Agent-Mediated Electronic Trading*, 1999.

[17] Karen K. Fullam, Tomas B. Klos, Guillaume Muller, Jordi Sabater, Andreas Schlosser, Zvi Topol, K. Suzanne Barber, Jeffrey S. Rosenschein, Laurent Vercouter, and Marco Voss. A specification of the agent reputation and trust (art) testbed: Experimentation and competition for trust in agent societies. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, 2005.

[18] Barbara J. Grosz. Colored trails: a formalism for investigating decision-making in strategic environments. In *In IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games*, 2005.

[19] Raz Lin, Sarit Kraus, Tim Baarslag, Dmytro Tykhonov, Koen Hindriks, and Catholijn M. Jonker. Genius: An integrated environment for supporting the design of generic automated negotiators. *Comput. Intell.*, 30(1):48–70, 2014.

[20] M. Evan Brooks. An annoted paiktograpgy ("game"ography). *Computer Gaming World: The Definitive Computer Game Magazine*, (75):12, 1990. Retrieved from http://www.cgwmuseum.org/galleries/issues/cgw_75.pdf.

[21] Angela Fabregues. personal contact via email. February 11, 2016.

[22] Sarit Krauss and Daniel Lehmann. Designing and building a negotiating automated agent. In *Computational Intelligence*, volume 11. 1995.

[23] Ari Shapiro, Gil Fuchs, and Robert Levinson. *Computers and Games: Third International Conference, CG 2002, Edmonton, Canada, July 25-27, 2002. Revised Papers*, chapter Learning a Game Strategy Using Pattern-Weights and Self-play, pages 42–60. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[24] D.E. Loeb and M.R. Hall. *Thoughts on Programming a Diplomat*. Université de Bordeaux 1/CNRS. Laboratoire Bordelais de Recherche en Informatique [LaBRI], 1990.

[25] Adam Webb, Jason Chin, Thomas Wilkins, John Payce, and Vicent Dedoyard. Automated negotiation in the game of diplomacy, 2008. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.466.6361&rep=rep1&type=pdf.

[26] João Ribeiro, Pedro Mariano, and Luís Lopes. Darkblade: a program that plays diplomacy. Master's thesis, Universidade de Aveiro, 2008.

[27] Fredrik Håård. Multi-agent diplomacy: Tactical planning using distributed problem solving. Master's thesis, Blekinge Institute of Technology, 2004.

[28] Stefan J. Johansson and Fredrik Håård. Tactical coordination in no-press diplomacy. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '05, pages 423–430. ACM, 2005.