A Generic Agent Architecture for Cooperative Multi-agent Games

João Marinheiro[1] and Henrique Lopes Cardoso[1,2]
[1]*DEI/FEUP, Faculdade de Engenharia, Universidade do Porto, Porto, Portugal*
[2]*LIACC – Laboratório de Inteligência Artificial e Ciência de Computadores, Porto, Portugal*

Keywords: Multi-agent Systems, Cooperative Games, Negotiation, Opponent Modeling, Trust Reasoning.

Abstract: Traditional search techniques are difficult to apply to *cooperative negotiation games*, due to the often enormous search trees and the difficulty in calculating the value of a players position or move. We propose a generic agent architecture that ensembles negotiation, trust and opponent modeling, simplifying the development of agents capable of playing these games effectively by introducing modules to handle these challenges. We demonstrate the application of this modular architecture by instantiating it in two different games and testing the designed agents in a variety of scenarios; we also assess the role of the negotiation, trust and opponent modeling modules in each of the games. Results show that the architecture is generic enough to be applied in a wide variety of games. Furthermore, we conclude that the inclusion of the three modules allows for more effective agents to be built.

1 INTRODUCTION

Games have always been an important testbed to develop new and interesting ideas in AI research. There has been extensive research using games like Chess (Drogoul, 1995) or Go (Silver et al., 2016). However, most research in this area relates to traditional adversarial games and makes use of extensive searching and game specific heuristics in order to find the best move[1].

One interesting category of games is that of *cooperative negotiation games*, where players are encouraged to barter and create or break deals between themselves in order to win. In the field of game theory, a game is considered a cooperative game if players are able to form binding commitments with eachother. Games in which players can not create binding agreements are usually considered non-cooperative games (Nash, 1951). It is usually assumed that communication between players is allowed in cooperative games but not in non-cooperative games.

For the purposes of this work, however, we consider a different, more general, definition of cooperative games – games in which cooperation between players is possible and encouraged but which is not necessarily based on binding agreements. More specifically, we will focus on cooperative *negotiation games* with a *mix of cooperation and competition*. Some characteristics that are frequently present in these games are:

- Very large search spaces, which makes the application of traditional search techniques impractical.
- Difficulty in evaluating moves and player positions due to how the values for these often depend on the *social context* of the game.
- The possibility of betrayals and desertions due to the existence of non-binding agreements.

Two examples of cooperative negotiation games are Diplomacy (Calhamer, 2000) and Werewolves of Miller's Hollow (des Pallières and Marly, 2009). Diplomacy is a strategy game for 7 players where each player takes control of a nation. By submitting orders to their armies and navies, which are executed simultaneously with those of other nations, players must try to capture territories and supply centers in a map of Europe. The first player to capture 18 or more supply centers wins the game. Because players can issue orders to support their opponents, the game encourages players to negotiate and cooperate among themselves in order to win the game.

Werewolves of Miller's Hollow, on the other hand, is a team based game where two teams, the villagers and the werewolves, attempt to eliminate each other.

[1]The DeepMind approach (Silver et al., 2016) is a late deviation to this status quo, and has recently been applied to imperfect information games (Heinrich and Silver, 2016).

Marinheiro J. and Lopes Cardoso H.
A Generic Agent Architecture for Cooperative Multi-agent Games.
DOI: 10.5220/0006253101070118
In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence (ICAART 2017)*, pages 107-118
ISBN: 978-989-758-219-6

The twist is that the players on the villager team do not know who their team mates are and who the werewolves are, encouraging players to communicate, share information and decide who to trust. The game is played in two phases: the day phase, where players communicate freely, ending with a vote to eliminate one player thought to be a werewolf by the remaining players; and the night phase, where players cannot communicate but can choose to use some special abilities depending on their role in the game.

Due to the characteristics of cooperative negotiation games it is possible to obtain much better results in these games if one is able to negotiate and coordinate with other players effectively. Unfortunately, while humans are very good at negotiation and intuitively distinguish who to trust, it is much harder for a computer to do so. In order to develop effective and believable software agents for these sorts of games, and cope with the large size of the search spaces, new strategies that can effectively tackle the concepts of negotiation, opponent modeling and trust reasoning need to be employed. We propose a generic agent architecture that tackles these concepts and facilitates the development of agents with these capabilities to play cooperative negotiation games.

The rest of this paper is structured as follows. In Section 2 we briefly describe some of the existing approaches in this area. Section 3 introduces the proposed architecture and its different modules. Then, in Section 4 we describe some agents implemented using the proposed architecture. In Section 5 we describe the tests and experiments executed using the developed agents, together with some results. Finally, in Section 6, we discuss the implications and conclusions taken from the results obtained as well as some possible future work towards the improvement of the proposed architecture and developed agents.

2 RELATED WORK

Cooperative negotiation games have not received too much attention in the field of artificial intelligence, the game of Diplomacy being a notable exception. We here discuss some of the many agents created to play Diplomacy, since to the best of our knowledge no significant research has been applied to other cooperative games.

One of the simplest agents implemented for Diplomacy – DumbBot – was developed by David Norman (Norman, 2013). This bot has no negotiation capabilities and uses a simple heuristic to decide its actions by preferring to choose moves that weaken its strongest opponents. DumbBot assigns a value to each territory that depends on who controls it and what units are around it, and then it assigns actions to every unit depending on those score values. While the method used is very simple, DumbBot obtains fairly good results and is frequently used as a benchmark for other Diplomacy agents.

One of the most important and influential negotiating agents is the Israeli Diplomat (Kraus et al., 1989; Kraus et al., 1995). This agent was designed with a general negotiation architecture to be applied in a variety of situations. The Israeli Diplomat tries to mimic the structure of a war-time nation. It consists of several components working together to choose the best course of action. The diplomat keeps a knowledge base of the relations it believes each nation has with each other, as well as any agreements it has and its intention to keep them, together with its trust that others will keep them as well. This knowledge base is updated by the different modules as the game progresses, and affects every decision the diplomat takes.

Another interesting approach in this area is that of D-Brane (de Jonge, 2015), an agent developed by Dave de Jonge that makes use of the NB^3 algorithm as well as a complex strategical module to find the best sets of moves to negotiate and play. This agent uses a basic form of opponent modeling by using the utility values of deals previously proposed and accepted by its opponents as a way to direct the search for better solutions; however, it does not make an attempt to explicitly predict an opponent's goals or strategies. Another aspect that this agent lacks is the ability to negotiate coalitions with other agents as well as joint moves for future phases of the game, having no negotiation strategy for these kinds of deals.

Finally, DipBlue (Ferreira et al., 2015) is a negotiating agent for Diplomacy inspired by the Israeli Diplomat architecture. The agent is split into several modules called Advisers, that together decide the actions the agent takes. Each adviser receives move scores evaluated by previous advisers and alters them according to its role. The base adviser is inspired by DumbBot and uses the same scoring heuristic. This score is then changed by other advisers to promote support actions for the units, promote actions that keep agreements with its allies and encourage the agent to attack players that it distrusts. In order to model the trust value of each player, DipBlue keeps a trust matrix that is updated as the game is played. If a player performs hostile actions against DipBlue, such as attacking or breaking an agreement, its trust value diminishes. If a player performs friendly actions, or refrains from doing hostile actions, its trust value increases. DipBlue is more likely to accept agreements and help players with which it has a high trust value,

and attack players with a low trust value. DipBlue does not have full negotiation capabilities, lacking the ability to ask and give information or threaten players.

3 ALPHA

This section introduces the proposed Alpha architecture and its modules, which intends to approach several of the challenges present in cooperative negotiation games and facilitate the development of agents for these games.

3.1 Alpha Architecture

We propose a generic architecture that tackles several of the issues present in cooperative negotiation games, allowing for the creation of agents capable of negotiation, trust reasoning and opponent modeling in a simple way. In order to be used in a variety of games and environments this architecture is required to be as generic as possible, making no assumptions about negotiation protocol, negotiation strategy, player goals, and search strategy. As such, the proposed architecture, which we dubbed the *Alpha architecture*, is modular and based on the architecture of the Israeli Diplomat and DipBlue. Figure 1 shows a simplified overview of the Alpha architecture.

Like the Israeli Diplomat, it is based on the structure of a wartime nation, with four different independent modules[2]:

- **The President:** in charge of coordinating other modules and taking the final decisions.
- **The Strategy Office:** in charge of suggesting good strategies to the President.
- **The Foreign Office:** in charge of negotiation.
- **The Intelligence Office:** in charge of predicting what opponents are likely to do.

This structure allows the architecture to have a clean separation between the different independent modules for the different subjects of negotiation, opponent modeling, the strategic and tactical evaluation of the game and the high level agent personality and overall strategy. As a result, the user can easily and effortlessly swap out the modules at will and decide what capabilities the agent should have, allowing for great freedom in designing agents. By creating a relatively small quantity of these modules and combining them in different ways the developer can create a large variety of agents with different behaviors and capabilities in a short time.

[2]These modules can also be seen as agents, making the Alpha architecture a multi-agent system, as in (Johansson and Håård, 2005; Johansson and Olsson, 2005).

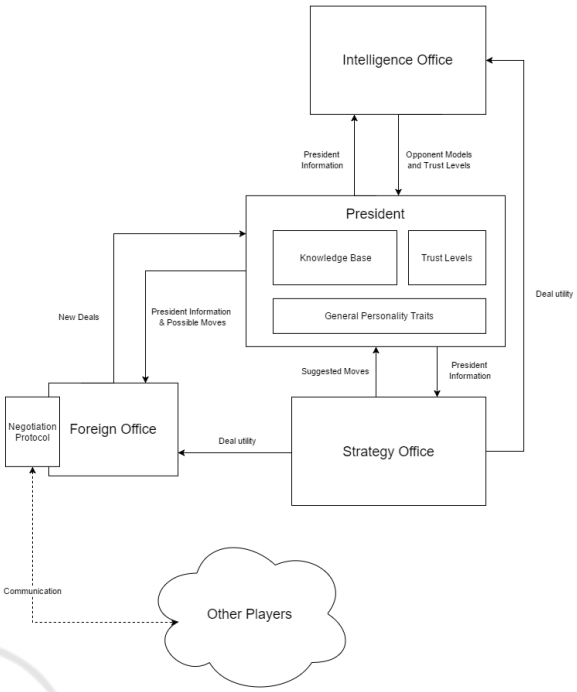

Figure 1: High-level diagram of the Alpha architecture and its modules.

3.1.1 The President

The President (PR) acts as the central module for the agent, holding its personality characteristics, coordinating the other modules, defining the overall high level strategy of the agent through the definition of its goals, and being in charge of selecting and executing the moves for the player.

In order to do this the PR keeps a knowledge base of everything it needs to know about the environment and its opponents. This knowledge base is then also used and modified by the remaining three modules, allowing the PR to make the best decisions possible with up-to-date useful information regarding the environment. The information contained in the knowledge base is the following:

- The current state of the game.
- The moves played during the course of the game by each player.
- The current player goals for the game and their importance.
- The lists of deals confirmed, completed and proposed by itself and other players over the course of the game.

- The player's current disposition towards other players, such as who are its allies and enemies.
- The opponent models for each other player.
- The general trustworthiness levels of each player.
- The trustworthiness levels of each deal.

Additionally, the PR also has a set of personality traits that can be defined depending on the game being played. These govern the general strategy of the PR, such as how aggressive it is, how trusting of other players it is or how prone to taking risks it is. Finally, the PR also keeps lists of moves and deals suggested by the other modules, which the PR can then choose to execute depending of several factors.

When defining the PR module the developer must define what constitutes a deal, a move and a goal because these are game-specific concepts that are difficult to generalize. Different games (such as Diplomacy and Werewolves of Miller's Hollow) have very different possibilities for what moves and deals a player can make. These definitions specify the game-specific elements that are stored in the knowledge base and used by all modules.

One important role of the President is deciding what overall goals the agent is striving for and the relative importance to attribute to each goal. This information is then used to inform the behavior of other modules, such as what moves are suggested by the Strategy Office or what deals are accepted by the Foreign Office. This allows the PR to dictate the overall strategy it wants to follow to its subordinate modules, allowing them to focus on the individual details of what actions are more likely to result in attaining these goals.

Different PR modules allow the developer to customize the agent's general strategy and personality allowing for different player archetypes (Ferreira et al., 2015).

3.1.2 The Strategy Office

The Strategy Office (SO) has the purpose of analyzing the strategical situation of the game and suggesting good moves to the PR. It contains most of the game-specific heuristics, evaluating the utility of possible moves and deals, and as such is highly dependent and adapted to the specific game being played. In addition, the SO also defines the search strategy used to explore the space of different moves.

This module is generally separated into two parts: the *search strategy* used when looking for moves to analyze and the *evaluation function* used for calculating the utility value of moves and deals. While the search strategy used can generally be applied to different environments relatively easily, the tactical evaluation is, in general, entirely dependent on the game being played, as it relies on specific knowledge about the game's rules in order to calculate what constitute good moves.

In order to find the best moves, the SO has access to the PR's knowledge base, being able to make use of the information contained there to evaluate the utility of different moves and deals. When the PR requests move suggestions, the SO finds good moves and suggests them back to the PR, who stores them in its internal list.

Swapping the Strategic Office allows a developer the choice among different search strategies and heuristics for the game, which can have a major impact on a player's effectiveness.

3.1.3 The Foreign Office

The Foreign Office (FO) has the purpose of managing any interaction with other players and negotiating deals and coalitions in a way that best allows the PR to execute the moves it is considering in order to fulfill its goals. When the PR requests the FO to negotiate with other players, it sends a list of which moves it is considering, for which the FO should attempt to find supporting deals. Using this information as well as any other information available in the PR's knowledge base that the FO finds useful, this module then autonomously communicates with other players and decides what deals to propose, reject and accept. When a deal is proposed, confirmed or completed the FO informs the PR so that it can add these deals to the appropriate lists in the Knowledge Base.

The *negotiation strategy* used is defined in this module and determines what deals are proposed and accepted and what concessions the agent is willing to make. This module also defines the *negotiation protocol* used by the agent when communicating with other players. The decision of what protocol to use is often dependent on the game being played or even the specific development framework on top of which the agent is being implemented.

Swapping the FO allows a developer to customize the negotiation capabilities of the agent, allowing the use of different negotiation and concession strategies, or even removing the FO altogether for an agent with no negotiation capabilities.

3.1.4 The Intelligence Office

The Intelligence Office (IO) is in charge of calculating the trust values and opponent models for the different players in the game.

This module is divided into two parts: the *opponent modeling* function and the *trust reasoning* function. The opponent modeling function outputs the predicted goals and their relative importance for each opponent, to be updated in the PR's knowledge base. How this is done is often specific to each game since the goals themselves as well as the actions and deals being analyzed are also specific to each game. The trust reasoning function is similar, outputting the trustworthiness values both for each opponent as well as for each individual deal, depending on how likely they are to be kept.

Different IOs can allow the developer to customize the opponent modeling and trust reasoning strategies, or lack thereof, of an agent. This module is especially useful in conjunction with the FO, since negotiations are likely to benefit from a good opponent model and accurate trust reasoning.

3.2 Alpha Framework

In order to simplify the application of the Alpha architecture, we developed a framework composed of several abstract classes which represent the modules and behavior described. These classes define what each module should do as well as the data available and how it is updated and communicated to and by each of the modules. Each module is defined in its own class and implements a specific interface.

In practice all a developer has to do to create an agent for a given domain is to implement the abstract classes of the modules he wishes to use. When implementing the modules, the developer must implement their abstract methods in order to define the domain specific negotiation strategies, protocols, heuristics, models and message handling. The data produced by the different modules is automatically updated and available for use by all modules for their calculations.

Additionally, the developer must also implement the data classes with the domain specific definitions of what a move, a deal, a goal and an opponent are. The different modules can then be attached to the PR, so that an agent can be instantiated to play the game.

4 THE ALPHA ARCHITECTURE IN PRACTICE

In this section we describe the details of the implementation of two different agents using the proposed Alpha architecture, for two very different cooperative negotiation games: Diplomacy and Werewolves of Miller's Hollow. These two games were chosen as a proof of concept to test the Alpha architecture due to the very different characteristics and challenges they offer as well as the wide breadth of existing work for the game Diplomacy.

4.1 AlphaDip

AlphaDip is a Diplomacy playing agent heavily based on D-Brane, using a modified version of its strategic module as well as the NB^3 algorithm to search for the best moves. It has a few key improvements compared with D-Brane, the most notable ones being an improved strategic module, a defined strategy for negotiating coalitions and some ability to predict its opponents' goals and trustworthiness.

In the context of AlphaDip, a move is considered a set of orders, one for each unit the player controls, as well as possible supporting orders from other players that may be necessary. The representation of player goals is relatively simple. Since in Diplomacy, the goal of the game is to capture as many supply centers as possible to win the game, a player's goals are simply a measure of how much they want to control a certain supply center at the moment. This is represented by a positive real number, where 0 means no intention to control a supply center, 1 means neutral intention to control a supply center and values above that meaning greater intentions of controlling a supply center. On the other hand,the trust values for players stored by the PR are positive real numbers that are inversely proportional to the trustworthiness of these players. This means that 0 represents full trust in a player, 1 represents neutral trust and values above that meaning lower levels of trust in a player.

When a round starts the PR first asks the SO for a fallback move that is expected to work even without any supporting deals with other players or any kind of negotiation. This move will be used by the PR in the eventuality that all negotiations fail, or in the absence of a FO. The PR then asks periodically the SO, who continually searches for the best moves, for suggested moves to consider.

AlphaDip's SO tries to find moves that maximize the number of controlled supply centers by the player, and is based on D-Brane's strategic module and the NB^3 algorithm. The objective of the game is to take control of as many supply centers as possible. As such, one way of determining the utility of a move is simply the number of supply centers that are ensured to be controlled by a player when it plays that move. This is the method used by D-Brane to calculate the utility of a move. AlphaDip calculates utility in a similar way, but introduces trust reasoning and the prediction of opponent goals in order to attempt to obtain a more accurate value than D-Brane. While

D-Brane attributes the same score of 1 to every supply center, AlphaDip uses the player goals to influence the value of each supply center. Additionally, if a move requires supporting orders from other players to succeed, their trust values are taken into account when determining the utility value. This is so that SO suggests moves that are likely to be easy for the FO to obtain any necessary supporting move commitments from other players. Equation 1 shows how the SO determines the utility of a move.

$$U_p(m) = \frac{\sum_{i=1}^{n} I_m(i) \times g_p(i)}{t_p(m)} \quad (1)$$

Where $U_p(m)$ is the utility value of move m for player p, n is the total number of supply centers, I_m is a function where for each supply center it returns 1 if that supply center is sure to be controlled after move m and 0 if not, g_p is the goal value that player p has or is assumed to have for each supply center, and t_p is the average trust that player p has on all other players involved in the move or 1 if no other players are involved.

The suggested moves may include order commitments by other players in order to be feasible and so the PR passes the current best move it is considering on to the FO for it to negotiate any possible support deals. AlphaDip's FO performs two types of negotiation: coalition establishment with other players and order commitments for the current round. This is an improvement on D-Brane, which did not have a strategy for the establishment of coalitions, instead assuming that all D-Branes simply formed a coalition against all other players in the game. Currently, AlphaDip is not able to negotiate move commitments for the following rounds as that would increase the complexity of the agent tremendously.

The strategy employed to negotiate coalitions is similar to the strategy used by DipBlue. Like DipBlue, at the start of the game AlphaDip proposes a peace agreement to every other player in the game. After that, during the rest of the game the FO attempts to propose alliances against the stronger player in the game with which it is not in peace with. If a player's trust value rises above a certain threshold (meaning the player is less trusted) the peace with that player is broken. Conversely, if the trust value drops below a certain level (meaning the player is trusted) AlphaDip proposes peace to this player. Additionally, if the game has 4 or less players remaining AlphaDip immediately breaks any alliances it has with a player if that player controls 14 or more supply centers. This is so that AlphaDip does not let a player get too close to winning in the final stages of the game.

The FO also attempts to negotiate joint order commitments for the current round. The PR periodically asks the FO to negotiate deals concerning the moves being currently considered by the PR. Each time this happens, the FO compares the utility of the suggested moves with the utility of each of the proposals it received and either accepts the best proposal received if it has more utility, or proposes any necessary joint order commitments required by the moves proposed by the PR.

If the FO receives a proposal from another player and it is compatible with any deals it has already accepted, it asks the SO to calculate the utility of that deal, and informs the President for it to store it in the proposed deals list of the knowledge base for later consideration. The reason it does not choose to immediately accept or reject the proposal, as explained in (de Jonge, 2015), is so that the SO is given some time to continue searching and looking for any possibly better options for other joint moves, before the agent commits to the proposed orders. By committing to an offer and adding it to the PR's confirmed deals list, the search performed by the SO is automatically constrained to only look for moves that satisfy the conditions in the accepted deals.

Finally, after a certain negotiation deadline has passed and depending on whether the FO managed to negotiate any supporting moves deemed necessary, the PR picks the best move suggested to it that has a chance to succeed.

AlphaDip may also use the IO to calculate trust values for players and predict their current goals in the game. In order to update the trust values, the IO uses a strategy similar to DipBlue (Ferreira et al., 2015), where trust in players increases steadily over the course of the game if no aggressive actions are taken by these players and decreases when aggressive actions are taken. The amounts by which the trust score increases and decreases are dependent on the current trust value the players have, as well as whether AlphaDip considers himself to be at peace or at war with these players. This way, if a player is highly trusted or in peace with AlphaDip, any aggressive actions it takes will have a bigger impact on that player's trust. On the other hand, if a player is not trusted or is at war with AlphaDip, any aggressive actions it takes have a smaller impact on that player's trust, since AlphaDip already expects that player to take aggressive actions.

The IO also attempts to predict its opponent's goals, that is, which supply centers it believes each player wants to control more, using a simple strategy exemplified in Algorithm 1. Each time a player takes an offensive action against a certain supply center, the

IO increases the likelihood that that player wants to control that supply center (line 9 in algorithm 1). If a player takes no offensive actions against a supply center, or takes actions that would help another player capture that supply center such as support orders, the IO decreases the likelihood that the player wants to control that supply center (lines 5 and 12). Like the trust values, these increases and decreases are dependent on the current values for each supply center. That way if a player is already expected to want control of a certain supply center any actions it takes have a small impact on the value for that supply center. On the other hand, if a player suddenly makes a move on a supply center that AlphaDip believed that player was not interested in, the value for that supply center will be affected more significantly.

Algorithm 1: AlphaDip IO Goal Prediction Algorithm.

```
1:  playerOpponents ← getOpponentsFromPR()
2:  opponentGoals ← getOpponentGoalsFromPR()
3:  for all op ∈ playerOpponents do
4:      for all g ∈ opponentGoals[op] do
5:          g ← g × 0.99
6:          forActions ← getActionsSupportingGoal(op,g)
7:          againstActions ← getActionsAgainstGoal(op,g)
8:          for all o ∈ forActions do
9:              g ← g + 0.04/g
10:         end for
11:         for all o ∈ againstActions do
12:             g ← g × 0.95
13:         end for
14:     end for
15: end for
16: return opponentGoals
```

4.2 AlphaWolf

Werewolves of Miller's Hollow was the second game chosen to test the implementation of the Alpha architecture. As there are, to the best of our knowledge, no available frameworks for the development of Werewolves of Miller's Hollow agents, we have implemented our own server using the JADE multi-agent framework. We then implemented an agent capable of communicating with this server and playing the game, which we nicknamed AlphaWolf.

In order to simplify the implementation, and because certain roles are more suited to be played physically with humans, we use a simplified version of the game with a subset of the available player roles and abilities. In our version of the game there are 4 possible roles for the players: werewolves, villagers, seers and doctors. Werewolves have the goal of killing every other non-werewolf player in the game while every other player has the goal of killing the werewolves. The werewolves, seers and doctors each have a special ability that they can secretly perform during the night phase of the game. Werewolves can collectively vote on an enemy player to kill during the night. The seers can choose any player to investigate during the night, learning its secret role. Finally, the doctors are able to choose a player, who if attacked by the werewolves during the night will be healed and remain in the game, informing the doctor that this happened.

The AlphaWolf's PR works much the same as described in Section 3. At the start of each phase of the game it requests the IO to update it's opponents' trust values and predicted goals. In the context of this game a player's goal is tied to its role and as such predicting a player's goal means predicting the likeliness that a player has a certain role. In the PR's knowledge base this means that each player is attributed a role certainty for each role, from 0 to 100%. The sum of all role certainties totals 100%, so that if a certain role has a certainty of 100%, the PR knows that player's role and consequently, its goals. A player's trust is represented by a positive real number, where 1 means neutral trustworthiness, 0 means no trustworthiness and values above 1 mean progressively higher trustworthiness.

Afterwards, the PR requests the SO to suggest a good move. What constitutes a move in this context is dependent on the current phase of the game but it always involves choosing a player, to either vote out of the game or as a target for the player's ability during the night phase. After a player is suggested by the SO, and depending on whether the current phase of the game allows negotiation between the players, the PR may ask the FO to attempt to negotiate with the other players for joint votes against some player or requests for investigation or healing. When the FO has finished negotiating, the PR decides to either vote or target a certain player for its special action. If no deal has been reached the player is randomly chosen from the list of players suggested by the SO, with players with higher utility being more likely to be picked, otherwise if a deal has been reached for a different player, the PR will take the action it agreed to on the deal.

The SO implements a simple strategy to suggest good potential players to either attempt to eliminate or protect, depending on the current phase of the game and the player's role and goals. The way it does this is by attributing to each player a threat score, which is a measure of what roles and goals the player believes an opponent to have (calculated by the IO) and how dangerous these roles are to the player. In general terms, if a player is on the werewolf faction, roles that have the ability to gather more information or use abilities that hinder the werewolves' actions will have

a higher threat level to the player. In the same way, if a player is on the villager faction, roles that have the ability to gather more information or hinder the werewolves are less likely try to kill the player, and thus are less threatening. Then depending on the current phase of the game and whether the actions available to the player will hinder an opponent, such as voting to kill it, or help another player, such as healing it, either this threat score is used as the utility for the move or its inverse is used, respectively.

Equation 2 shows how the threat value of a player is calculated.

$$T_p = \frac{\sum_{i=1}^{n} B_i \times C_i}{\sum_{i=1}^{n} B_i} \quad (2)$$

Where T_p is player p's threat value, n is the number of different possible roles a player can be, B_i is the base threat value for role i and C_i is the current certainty that player p has role i.

The SO also has the purpose of calculating the utility of deal proposals. This utility is based on the previously mentioned threat value of the proposer of the deal, the threat value of the player whom the deal concerns, what type of action the deal is proposing and the trust of the player in the proposer of the deal. This calculation is described in Algorithm 2. The type of action proposed and the threat values for the proposer and the player affected by the proposal are used to calculate two values, one for the proposer and one for the target of the proposed action, representing how much the player is willing to help the proposer and hurt the target. These two values are then multiplied together with the trust on the proposer, representing how much the player trusts the proposer to abide by the deal and not take any actions against him, to reach the final utility value for the deal (line 9).

Algorithm 2: AlphaWolf SO Deal Utility Calculation.

1: target ← getTargetFromDeal()
2: proposer ← getProposerFromDeal()
3: proposerValue ← $\frac{1}{getThreatValue(proposer)}$
4: **if** dealActionIsPositive() **then**
5: targetValue ← $\frac{1}{getThreatValue(target)}$
6: **else**
7: targetValue ← getThreatValue(target)
8: **end if**
9: **return** targetValue × proposerValue × getTrustFromPR(proposer)

In Werewolves of Miller's Hollow the players can only communicate during certain phases of the game, namely the discussion phase and, for werewolves, the night phase. As the game is very reliant on communication between players negotiation is very important in order to obtain effective players. Otherwise players would not be able to coordinate their votes or use their abilities during the night. This is the purpose of the FO.

AlphaWolf's FO implements a simple negotiation strategy where each player proposes a joint vote against another player who they think is the most threatening, as well as other agreements such as investigation or heal requests depending on their levels of trust with other players. Players then await for confirmations from their opponents that they'll follow the agreement, locking the agreement in place if they receive a confirmation. Each negotiation round players compare the utility of the proposals they received with their concession value, which is based on their own proposal and decreases over time, and decide either to continue waiting or accept another proposal, retracting their own.

AlphaWolf's IO has the function of attempting to predict a player's goals and its trustworthiness. As described previously, since a player's goals are tied to its role in the game predicting its goals is a matter of predicting its role. In order to predict an opponent's role the IO analyses the proposals and votes of that player over the course of the game. The predicted role certainties for that opponent are thus a function of the threat values of the players that that opponent voted against, or proposed votes against, and the rounds in which that player took those actions. Algorithm 3 describes the calculation for the prediction of opponent goals by the IO.

The IO searches through each player's past actions and for each vote or proposal that that player made it calculates a vote or proposal damage value (line 9 in algorithm 3). This value indicates the likeliness that an action was taken with the intent of damaging the player's faction and is based on the threat values of the players who are the targets of that opponent's actions. A high threat value for the target of the action indicates that the action was not very damaging to AlphaWolf's faction, and may have even been helpful, and a low threat value indicates a damaging action, as that opponent was voting against players that are considered likely to be allies.

For each action its vote damage is then used to calculate a scaling factor for each possible opponent role (line 10) and that scaling factor is finally used to scale the role certainties of each role proportionally (lines 11-12). The reason each role has a different scaling factor is because certain roles are more likely to have more information than others. So seers have a higher scaling factor than doctors, and doctors have

a higher scaling factor than villagers. In this way if an opponent takes a damaging action, the likeliness that it is a seer diminishes comparatively more than the likeliness that it is a villager, since a seer would be less likely to commit damaging actions against the villagers, having more information about the player roles. These scaling factors are then multiplied with each current role certainty, and the values for the roles are then re-scaled back so that they total 100% (line 14).

To calculate the trust values the IO analyses the previous round and checks for each opponent if it kept any agreements it accepted or if it voted against the player. If an opponent broke an agreement or voted against the player its trust value is decreased by a certain amount, otherwise its trust value increases.

Algorithm 3: AlphaWolf IO Role Prediction Calculation.

```
1:  pastRounds ← getPastRoundsFromPR()
2:  opponents ← getOpponentsFromPR()
3:  opponentGoals ← getOpponentGoalsFromPR()
4:  for all op ∈ opponents do
5:      for all round ∈ pastRounds do
6:          roundAgeFactor ← getAgeFactor(round)
7:          for all actions ∈ getOpponentActions(op, round)
            do
8:              target ← getTarget(action)
9:              voteDamage ← getThreatValue(target) / getAverageThreatValue() × roundAgeFactor
10:             scalingFactors ← calculateRoleScalingFactors(voteDamage)
11:             for all r ∈ opponentGoals[op] do
12:                 r ← r× scalingFactors[r]
13:             end for
14:             normalizeOpponentGoals(opponentGoals[op])
15:         end for
16:     end for
17: end for
18: return opponentGoals
```

5 EXPERIMENTS AND RESULTS

This section describes the methodology and results of the tests and experiments performed with the agents implemented using the Alpha architecture.

5.1 AlphaDip Tests Configuration

The tests on AlphaDip were conducted using the DipGame platform (Fabregues and Sierra, 2011). We compared AlphaDip with two previously developed Diplomacy playing agents, DumbBot and DipBlue[3].

[3]These were chosen mainly for their availability in the DipGame platform.

In order to compare the performance of AlphaDip with those agents we performed some of the same tests reported in (Ferreira, 2014) and (de Jonge, 2015) and compared our results with the results obtained there. It should be noted that unlike the tests performed using D-Brane in (de Jonge, 2015), which assumed that the D-Branes always formed a coalition against every other agent in the game, we allow our agents to negotiate at will, establishing and breaking coalitions.

In each of the experiments, we tested AlphaDip using 3 distinct configurations, in order to separately test the impact of negotiation and opponent and trust modeling in its performance. The configurations used were: an agent using only the PR and SO, an agent adding to this the FO and, finally, an agent using all 4 modules: PR, SO, FO and IO.

For every configuration, in each experiment we played a number of games of Diplomacy, stopping after 40 game phases. After a game had finished we ordered the players by ranking, from 1st to 7th, and collected the ranking results. Ranking is determined by the number of supply centers that a player controlled at the end of the game if that player was alive, or by the game phase in which that player was eliminated otherwise. Players with more supply centers or that were eliminated later than other players have a higher rank in the game and are thus considered better. For configurations without the FO (and thus, with no negotiation capabilities) we played a total of 100 games. For configurations having the FO we set the negotiation deadline at 15 seconds per round. Since these tests take considerably longer to execute we only played a total of 50 games per configuration in these cases.

All tests and experiments were performed in a laptop computer with 8GB of RAM and an Intel Core i5-6440HQ mobile CPU clocked at 3.5GHz.

5.2 AlphaDip Results

Like in Ferreira's (Ferreira, 2014) and de Jonge's (de Jonge, 2015) works, who had two instances of their agents play against 5 DumbBots, we had two instances of AlphaDip play against 5 instances of DumbBot. Since we have two agents playing in each game, the best possible average rank for our agent in these tests is 1.5, while the worst possible average rank is 6.5. By comparing the average ranks of AlphaDip with the average ranks of DipBlue and D-Brane we can determine how well each agent performs comparatively against the DumbBot. The best rank achieved by DipBlue in the tests performed by Ferreira is 3.57 (Ferreira, 2014),

Table 1: The average rank of 2 AlphaDips when playing with 5 DumbBots.

AlphaDip Config.	Avg Rank
PR + SO	2.35 ± 0.15
PR + SO + FO	2.21 ± 0.21
PR + SO + FO + IO	2.11 ± 0.19

Table 2: The average rank of 2 AlphaDips when playing with 2 DipBlues and 3 DumbBots.

AlphaDip Config.	AlphaDips	DipBlues
PR + SO	2.38 ± 0.16	5.03 ± 0.20
PR + SO + FO	3.56 ± 0.26	3.44 ± 0.32
PR + SO + FO + IO	2.3 ± 0.22	4.73 ± 0.29

while the best rank obtained by D-Brane in the tests performed by De Jonge is 2.35 (de Jonge, 2015).

Table 1 shows the average rank obtained by AlphaDip in each configuration and their standard deviations. These results show that AlphaDip plays significantly better than the DumbBot and DipBlue, even without having negotiation, opponent modeling or trust modeling. However it also appears that the inclusion of the IO and the FO only has a small effect on the performance of the agent. A t-test performed on the data from these tests obtains a value of 0.554 when comparing the results of the second test with the last test, and a value of 0.109 when comparing the first test with the last. This second value indicates that the difference in the second case appears to be statistically significant, though further testing would be required to confirm the significance of the increase in performance of the AlphaDips with all modules active compared to those having just the PR and SO.

In order to complement the previous experiments we also tested AlphaDip in an environment with a higher number of negotiating agents, having 2 AlphaDips play with 2 DipBlues and 3 DumbBots. The 2 DipBlues played with the standard adviser configuration described in (Ferreira, 2014) in all tests. The results for this experiment are shown in Table 2.

These results show that when the AlphaDips are playing with just the PR and SO or with all modules running they get a similar average rank between 2.3 and 2.4. A statistical t-test performed using the data from these tests gives us a value of approximately 0.653, which shows that the difference observed is not statistically significant. This indicates that the inclusion of negotiation and trust reasoning does not significantly affect the performance of the AlphaDips. One exception is when they play with the FO but without the IO, meaning that they negotiate without making any attempt to predict their opponent's goals or trustworthiness, where their average ranking decreases significantly to 3.56.

This lack of impact from negotiation is similar to the results obtained by De Jonge in (de Jonge, 2015), where he concludes that even though the NB^3 algorithm manages to find good joint moves for the players when they exist, the impact of these in the overall result of the game is negligible. As such, negotiating joint moves only for the current round is not enough to significantly increase the performance of the players – in order to obtain better results one would have to attempt to negotiate further rounds ahead as well.

An interesting observation is that when playing with the DipBlues, while the AlphaDips do obtain a small increase in their average rank when all modules are running compared to having just the PR and SO, the DipBlues themselves are also benefited. The DipBlues obtain a better average ranking of 4.73 when playing with the AlphaDips with all modules active, compared with an average ranking of 5.03 when playing with AlphaDips incapable of negotiation. These observations were also corroborated by De Jonge's observations in (de Jonge, 2015), where he found that other agents could also benefit from the deals discovered by agents running the NB^3 search algorithm.

5.3 AlphaWolf Tests Configuration

Tests were also run using the Werewolves of Miller's Hollow playing agent, AlphaWolf. Unfortunately, due to the lack of any comparable agents developed for this game, we were only able to test the relative performance of our agent with different configurations of active modules.

In order to test the effect of the different modules on the performance of the agents we opted to have the werewolves always playing with all modules running, and changed only the configurations of any villager agents in the different tests. This way we can easily see the effect that each module has on the performance of the villagers. We tested 3 different configurations for this agent, just like with AlphaDip: just the PR and the SO, all modules except the IO and all modules active.

In each test we had the agents play 100 games in a 10 player game where 2 of the players were werewolves, and the remaining 8 were from the villager faction, with 1 seer, 1 doctor and the remaining 6 being standard villagers. This ratio of werewolf players to villager players was chosen because it is the recommended ratio for werewolves to villagers in the official Werewolves of Miller's Hollow rules. We recorded the win percentage for the villagers over those games as well as the average number of villager agents left alive at the end of the game when the villager faction won.

Table 3: The win percentages and average number of remaining villagers for a team of 8 villagers playing against 2 werewolves.

Villagers Config.	Win %	Avg # Villagers
PR + SO	27%	3.56
PR + SO + FO	46%	3.20
PR + SO + FO + IO	73%	4.88

5.4 AlphaWolf Results

Unlike with Diplomacy, the results for the AlphaWolf agents, shown in Table 3, show that negotiation, trust modeling and opponent modeling have a significant impact on the effectiveness of the agents. With the inclusion of the FO and the IO the performance of the agents steadily increases from a 27% win ratio to a 46% win ratio, and finally a 73% win ratio with all modules active. The inclusion of the IO also significantly increases the number of villagers remaining alive in games where the villagers win. This means that with the inclusion of trust and opponent modeling the players are able to identify the werewolves much earlier in the game, allowing for quicker victories.

One possible explanation for the difference in the relative impact of negotiation, trust and opponent modeling in the game of Werewolves of Miller's Hollow compared with Diplomacy is that in the latter the agents already implicitly have an idea of their opponent's goals. Since in Diplomacy the rules of the game encourage players to capture supply centers, players can already play with the assumption that other players will try to maximize their number of supply centers over the course of the game. On the other hand, in the game of Werewolves of Miller's Hollow players have no way to know at the start of the game what their opponents will be trying to do. This means that players have no way to predict an opponent's utility function at the start of the game, and must analyze a player's actions in order to predict it.

Negotiation may also have a greater impact in Werewolves of Miller's Hollow because each player only has a single vote to affect the round. Without coordination and organizing joint votes, players have a hard time completing their goals. In Diplomacy players can have differing numbers of supply centers and units, which allows certain players to affect the outcome of the rounds more than others; this means strong players can use their superior strength to obtain their objectives even without negotiation.

6 CONCLUSIONS AND FUTURE WORK

The objectives of this work were the study of what elements were necessary to create effective agents that could play cooperative negotiation games, and development of a generic architecture that included these elements that could be used to facilitate the development of effective agents for a wide variety of games.

We tested the proposed architecture by developing agents for two very different cooperative negotiation games and believe that the proposed Alpha architecture is generic enough to be applied to many other different games. The two most important agents developed using the Alpha architecture and framework were AlphaDip and AlphaWolf. AlphaDip is an agent with strategies inspired by D-Brane and DipBlue, with the inclusion of opponent modeling to make predictions about an opponent's intention to capture certain supply centers, as well as a negotiation strategy for the establishment of coalitions, which was not present in D-Brane. AlphaWolf is a Werewolves of Miller's Hollow agent that also includes negotiation, trust and opponent modeling capabilities, allowing it to predict its opponents' roles and negotiate deals accordingly.

The results of the tests performed using these two agents show that AlphaDip was in general superior to both DipBlue and D-Brane, obtaining better average ranks in the games played. However, the inclusion of negotiation, trust reasoning and opponent modeling capabilities did not have a very large impact on the performance of the agent. The results obtained for AlphaWolf show that the inclusion of the Foreign Office and Intelligence Office had a larger impact in the performance of the agent. This indicates that negotiation, trust and opponent modeling are more important in Werewolves of Miller's Hollow than in Diplomacy.

We believe that these results are positive and the inclusion of negotiation, trust reasoning and opponent modeling capabilities generally improved the performance of the agents, though the impact was much greater for AlphaWolf than for AlphaDip. We also believe that the developed architecture and framework are a helpful contribution to the field by facilitating the development of agents with these capabilities.

However, while the developed architecture is very modular and allows agents to be built upon it and make use of negotiation, trust and opponent modeling, there is still room for improvement. The Alpha architecture allows developers to define the different strategies for negotiation, trust reasoning and opponent modeling that they want to use depending on the specific game, such as D-Brane's NB^3 algo-

rithm. However, this process can be made simpler by the inclusion of generic strategies that can be applied equally to any environment. The inclusion of a generic way to predict opponent goals and strategies, calculate trust values and decide what deals to accept, based on the knowledge base of the President, would simplify the process of developing an efficient agent even more.

An even larger step in obtaining a truly generic system would be the inclusion of some form of abstract understanding about the rules of the game being played and the board state, which could be defined by the developer using a formal language such as the one used in the Zillions Of Games software (Corporation, 2016) or in the General Game Playing project (Genesereth et al., 2005). With this capability it would be possible to have a system that could generate agents able to play and negotiate in many different types of negotiation games, by simply providing it with a file containing an abstract description of the game.

The agents implemented during the course of this work, while generally efficient, could also be improved. One major improvement to AlphaDip could be to allow the agent to search for and negotiate movement commitments for several rounds ahead instead of only the current round. In the case of AlphaWolf and the Werewolves of Miller's Hollow server implemented, a key improvement would be the capability for AlphaWolf to use strategies involving bluffing, by for example making opponents believe it has a different role than its true role, a strategy human players frequently use in the game. If correctly implemented, this ability could make AlphaWolf much more effective, especially when playing with human opponents.

ACKNOWLEDGEMENTS

We wish to thank André Ferreira and Dave de Jonge for their previous work in this area, upon which our work is based, as well as always being available to talk and help whenever we needed.

REFERENCES

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

Corporation, Z. D. (2016). Zillions of Games. http://www.zillions-of-games.com/. Accessed: 23-11-2016.

de Jonge, D. (2015). *Negotiations over Large Agreement Spaces*. Phd thesis, Universitat Autonòma de Barcelona.

des Pallières, P. and Marly, H. (2009). Werewolves of Miller's Hollow: The Village. Lui-même.

Drogoul, A. (1995). When ants play chess (or can strategies emerge from tactical behaviours?). In Castelfranchi, C. and Müller, J.-P., editors, *From Reaction to Cognition: 5th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW '93 Neuchâtel, Switzerland, August 25–27, 1993 Selected Papers*, pages 11–27. Springer.

Fabregues, A. and Sierra, C. (2011). DipGame: A challenging negotiation testbed. *Engineering Applications of Artificial Intelligence*, 24(7):1137–1146.

Ferreira, A. (2014). *Dipblue: a diplomacy agent with strategic and trust reasoning*. Master thesis, Universidade do Porto.

Ferreira, A., Lopes Cardoso, H., and Reis, L. P. (2015). Strategic negotiation and trust in diplomacy – the dipblue approach. In Nguyen, N. T., Kowalczyk, R., Duval, B., van den Herik, J., Loiseau, S., and Filipe, J., editors, *Transactions on Computational Collective Intelligence XX*, pages 179–200. Springer International Publishing, Cham.

Genesereth, M., Love, N., and Pell, B. (2005). General Game Playing: Overview of the AAAI Competition. *AI Magazine*, 26(2):62–72.

Heinrich, J. and Silver, D. (2016). Deep reinforcement learning from self-play in imperfect-information games. In *Proceedings of the Third Deep Reinforcement Learning Workshop, NIPS-DRL*.

Johansson, S. and Olsson, F. (2005). Mars – a multi-agent system playing risk. In *Proceedings of Pacific Rim International Workshop on Multi-agents (PRIMA)*.

Johansson, S. J. and Håård, F. (2005). 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, New York, NY, USA. ACM.

Kraus, S., Gan, R., and Lehmann, D. (1995). Designing and Building a Negotiating Automated Agent. *Computational Intelligence*, 11(972):132–171.

Kraus, S., Lehmann, D., and Ephrati, E. (1989). An automated diplomacy player. In Levy, D. and Beal, D., editors, *Heuristic Programming in Artificial Intelligence: The 1st Computer Olympiad*, pages 136–153. Ellis Horwood Limited, Chinester, UK.

Nash, J. (1951). Non-Cooperative Games. *The Annals of Mathematics*, 54(2):286–295.

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

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503.