ANAC 2018 Diplomacy Challenge
Manual and Rules

Dave de Jonge

May 4, 2018

Abstract

This document describes the setup of the ANAC 2018 Diplomacy Challenge and explains how to implement an agent to participate.

Update: The deadline to submit your code has been extended to **21 May 2018 23:59 UTC-12**.

1 Introduction

Diplomacy is a strategy game for 7 players involving multilateral negotiations. The main motivation to use Diplomacy as a domain for ANAC is that in Diplomacy, unlike most other existing negotiation domains, the utility functions are not directly given in the form of a closed-form expression. Instead, the agents need to apply heuristics to estimate the value of any deal and one cannot expect an agent to ever have perfect knowledge of its utility function. We therefore think that this domain is more challenging and more realistic than most other negotiation domains.

2 Goal of the Challenge

The goal of this challenge is to implement a negotiation algorithm on top of an existing Diplomacy playing agent. The winner of the challenge will be the algorithm that outperforms all other participants as well as the non-negotiating agent.

Since the game of Diplomacy is a very challenging domain, we have decided to have it this year as a *Challenge* rather than a *Competition*. This means that we will only award a prize if the winner manages to outperform the non-negotiating agents.

The game of Diplomacy involves more than just negotiations. However, since the ANAC is only about negotiations, participants should *not* implement a full Diplomacy player from scratch. Instead, **you should only implement a negotiation algorithm, which will be used on top of an existing Diplomacy playing agent**. In fact, you are not even allowed to alter anything about the agent other than its negotiation algorithm.

Participants in the Diplomacy Challenge are expected to be familiar with the Java programming language.

3 Diplomacy

In this section we give a short overview of the game of Diplomacy. This is only meant to give you a global idea of the game, but does not serve as a complete introduction. If you are not familiar with the rules of Diplomacy, then please take a look at the complete rulebook:
`https://www.wizards.com/avalonhill/rules/diplomacy.pdf`.
Additional rules specific to the ANAC Diplomacy Challenge are discussed in the next section.

Diplomacy is a widely played game for 7 players, with no chance moves and no hidden information.[1] Players make their moves simultaneously. It is designed in such a way that each player needs to negotiate with the other players in order to have a chance of winning. It can be played as a classical board game, or it can be played online.[2]

The game takes place on a map of Europe in the year 1901, which is divided into 75 *Provinces*. Each player plays one of the 7 great *Powers* of that time: *Austria*, *England*, *France*, *Germany*, *Italy*, *Russia* and *Turkey* and each player starts with 3 units (also called armies or fleets) that are placed on the map (except Russia, which starts with 4 units). Some of the Provinces are so-called *Supply Centers*. There are 34 Supply Centers and the goal of the game is to conquer 18 or more of these Supply Centers.

The game iterates through five types of rounds (or 'phases'), in the following order:

- Spring phase
- Summer phase

[1] One might say that Diplomacy does have hidden information, because players make secret agreements. However, these agreements have no formal meaning, and form part of the players' strategies rather than of the rules of the game. Therefore, *formally* speaking there is no hidden information.

[2] http://www.playdiplomacy.com/

- Fall phase
- Autumn phase
- Winter phase

The first round of the game is referred to as Spring 1901, followed by Summer 1901, Fall 1901, Autumn 1901, Winter 1901, Spring 1902, Summer 1902, etcetera. In each round all players simultaneously 'submit orders' for all of their units.

3.1 Spring and Fall phases

During Spring and Fall phases each player must submit for each of his or her units one of the following three types of orders:

- A *move-to* order, meaning that the unit will try to move from the province where it is currently located to an adjacent province.
- A *hold* order, meaning the the unit intends to stay in the same province where it already is.
- A *support* order, also meaning that the unit will not move, but in this case it will also give extra strength to another unit.

If two or more units are ordered to move to the same province, then only the unit that receives the most successful supports will succeed and is said to *conquer* that province. A support order is unsuccessful, however, if there is another unit moving to the location of the supporting unit. In that case we say the support has been *cut*.

3.2 Summer and Autumn Phases

If, during a Spring or Fall phase, a unit tries to stay in a province, but another unit successfully conquers it, we say the first unit is *dislodged*. This means that during the following Summer or Autumn phase the unit must retreat to another province adjacent to its current location. Units that are not dislodged don't do anything during a Summer or Autumn phase.

3.3 Winter Phases

If a player successfully moves into (or stays in) a Supply Center during a Fall phase that player becomes the *owner* of the Supply Center. That player stays the owner of that Supply Center until any other player manages to occupy it during a Fall phase.

During Winter phases each player may be allowed to build more units, or may be obliged to remove some of his or her units from the map. Specifically, if a player owns m Supply Centers and currently has n units, with $m > n$, then he or she can build $m - n$ new units. On the other hand, if $m < n$ then he or she must remove $n - m$ units.

3.4 End of the Game

A player is eliminated when he or she loses all his or her units. A player wins the game when he or she owns 18 supply centers (a *Solo Victory*), but a game may also end when all surviving players agree to a draw. Any player may propose a draw when he or she likes to.

3.5 Negotiations

The main difference between Diplomacy and other deterministic games like chess and checkers, is that in Diplomacy players are allowed to negotiate with each other and form coalitions. Each round, before the players submit their orders, the players get some time to negotiate with each other and make agreements about the orders they will submit. Negotiations take place in private, and agreements that are made are only known to the players who are involved in the agreement. Players that are not involved in it are not aware of the agreement.

Typically, players may agree not to invade certain provinces, or they may agree that one player will use some of his or her units to support a unit of the other player. In this way, players essentially form coalitions against other players. These coalitions are not given beforehand. During the course of the game players may form coalitions and break coalitions as they like.

In a real Diplomacy game there are no formal rules for the negotiations. Players are allowed to negotiate anything and there is no guarantee that they will indeed obey the agreements they make. However, **in the ANAC Diplomacy Challenge there is only a limited set of deals that the players can make**, which we will discuss in the next section, and **the players will always obey their agreements.**

4 Diplomacy for ANAC

4.1 Framework

The competition will be run on the BANDANA framework:
`http://staff.scem.westernsydney.edu.au/~dave/bandana`
which is an extension of the DipGame framework. The BANDANA framework provides all the necessary Java classes to run a Diplomacy tournament (except for the Parlance game server) and analyze the results. Furthermore, it provides the Java classes required to implement a negotiating Diplomacy agent.

4.2 The Negotiation Protocol

Negotiations will only take place during Spring and Fall phases. The agents do not negotiate according to the Alternating Offers protocol or any other turn-taking protocol. Instead, we will use the *Unstructured Negotiation Protocol* [1]. This means that each agent may propose any deal whenever it wants. A deal may involve any number of agents, as long as it involves at least one agent other than the proposer. Once all players involved in the deal have accepted it, a special *Notary* agent checks whether it is consistent with earlier made agreements. If this is indeed the case then the Notary will send a confirmation message to all agents involved in the deal. The deal is only considered officially binding once the Notary has sent this confirmation message. Players may make as many proposals as they wish, and may continue negotiating after a deal has been confirmed.

Note that a proposal is only sent to the players that are involved in the deal, so the other players will not be aware of the fact that this deal has been proposed. Also, the Notary sends its confirmation message only to the players involved in the deal, so the agreement remains secret.

4.3 Allowed Proposals

As explained, in real Diplomacy there are no formal rules for the negotiations. However, for the ANAC Diplomacy Challenge there is a well-defined restricted set of proposals the players can make.

The kinds of deals the agents may propose to one another are those deals that are represented by a BasicDeal object in the BANDANA framework. That is: a deal may consist of any number of Order Commitments (a promise to submit a certain order during a specified phase) and any number

of Demilitarized Zones (an agreement between a set of players that neither of them will move into a given set of provinces during a specified phase).

In the ANAC Diplomacy Challenge negotiations will only take place during each Spring or Fall phase, and the Order Commitments and Demilitarized Zones may only refer to orders during Spring and Fall phases.

Definition 1. *An **Order Commitment** oc is a tuple: $oc = (y, \phi, o)$, where y is a 'year' (an integer greater than 1900), $\phi \in \{Spring, Fall\}$ and o is any legal Move-to order, Hold order, or Support order.*

An order commitment is a promise that a power will submit a certain order during a certain phase and year. For example: *"The army in Holland will move to Belgium in the Spring of 1902"*. However, if the order is a HLDOrder, then the corresponding power is still allowed to submit a SUPOrder or SUPMTOOrder for that unit instead of the HLDOrder.

Definition 2. *A **Demilitarized Zone** dmz is a tuple: $dmz = (y, \phi, A, B)$ with y and ϕ as above, A a nonempty set of Powers:*

$$A \subset \{Austria, England, France, Germany, Italy, Russia, Turkey\}$$

and B a nonempty set of Provinces.

A Demilitarized Zone consists of a phase, a year, a set of Powers and a set of Provinces, with the interpretation that none of these Powers is allowed to enter (or stay inside) any of these Provinces during that phase and year. For example the Demilitarized Zone $(1903, Fall, \{FRA, GER, ENG\}, \{NTH, ECH\})$ has the interpretation *"In the Fall of 1903 FRA, GER, and ENG will not enter the North Sea and will not enter the English Channel"*.

Definition 3. *A **Deal** d is a set:*

$$d = \{oc_1, \ldots oc_n, dmz_1, \ldots dmz_m\}$$

where each oc_i is an Order Commitment, each dmz_i is a Demilitarized Zone, and with $n \geq 0$, $m \geq 0$, and $d \neq \emptyset$.

A proposed deal can only be accepted or rejected in its entirety. It is not possible to only accept part of the deal. In the case you wish to accept only a part of the deal, you simply need to propose a new deal which only consists of those Order Commitments and Demilitarized Zones you wish to include in it. For example, if Austria proposes a deal $d = \{oc_1, oc_2\}$ to Germany, then Germany can choose to either accept d or to reject d, but

cannot choose to only accept $\{oc_1\}$. Instead, however, Germany can decide to make a new proposal $d' = \{oc_1\}$ to Austria, so then it is up to Austria to determine whether to accept or reject d'.

Apart from proposing this type of deals, your agent is also allowed to propose a draw to all the other surviving players, by calling the proposeDraw() method of the ANACNegotiator class. The game ends in a draw if all agents that have not been eliminated propose a draw in the same round of the game (see Section 'Proposing Draws' of the BANDANA manual).

For more information about the negotiation protocol we refer to the BANDANA manual.

4.4 Commitments

In a real Diplomacy game the agreements made between the players do not have any formal consequences and players may break their promises. In the ANAC Diplomacy Challenge, however, your agent will be based on the *D-Brane tactical module* (see the BANDANA manual) which will choose the orders your player will submit, and which will do this in such a way that it always obeys all agreements that your negotiation algorithm has made. Therefore, when your agent makes a deal, you can be sure that all agents involved in the deal (including your own agent) will indeed respect this commitment.

5 Tournament Setup

The ANAC Diplomacy Challenge is composed of two rounds.

5.1 First Round

In the first round 3 or 4 instances of your agent will be playing a large number of games against 3 or 4 instances of the non-negotiating agent. This agent will play be identical to your own agent, except that it does not negotiate. Your agent passes the first round if the instances of your agent achieve a higher average score than the non-negotiating agents. If no participant passes the first round, then the Challenge ends with no winner.

We will make the non-negotiating agent available for you so that you can already test for yourself whether the agent will pass the first round or not, before submitting the agent. More details about this will follow later. If you think your agent will not pass the first round, we still recommend you to submit your agent.

5.2 Second Round (if there are at least 2 agents that passed the first round)

In the second round all agents that passed the first round will be playing together in a large number of games. In each of these games there will be a maximum of 4 negotiating agents. The other agents will be non-negotiating.

The negotiating agent that scores the highest number of points in this round wins the challenge. If multiple agents score equally, then the results of the first round will be used as a tiebreaker. If we then still have multiple participants with equal score, then the prize will be divided among them.

5.3 Second Round (if your agent is the only agent that passed the first round)

In this case your agent will also be playing with 3 randomly selected other agents even though they did not pass the first round (if there are not enough submissions then we may use the agents from last year's competition). If none of the other agents achieves a higher score than your agent, then you win the challenge. Otherwise, the challenge ends with no winner.

5.4 Statistics

In the above, the score is always an average over a large number of games. We will play as many games as necessary to obtain statistically significant results. Furthermore, whenever we say "higher score" we mean that the difference in score is statistically significant. Whenever we say "equal score" we mean that the difference in score is so small that it is not statistically significant.

5.5 Motivation

The motivation for this tournament setup is that the first round ensures that the players do not make purely selfish proposals. They have to be able to make proposals that are beneficial to the coalition as a whole. On the other hand, the second round ensures that the agents cannot just accept any random proposal. An agent that accepts anything would be easily exploited by its opponents in the second round. Furthermore, the second round provides your agent the opportunity to try and exploit the other players.

5.6 Other details

- Each round of the game will last for 30 seconds. It is the participants own responsibility that their players' negotiation algorithm finishes before that deadline.
- The "score" of an agent is simply the average number of Supply Centers it conquers.
- A game ends either if any player conquers 18 Supply Centers, or when the players agree to a draw, or if the game reaches the 'winter 1920' phase.
- In each game the agents are randomly assigned to any of the 7 Great Powers.

6 How to Implement your Agent

To participate in ANAC Diplomacy Challenge you must implement a Java class which extends the `ANACNegotiator` class which is provided with the BANDANA framework. The BANDANA framework and its manual can be downloaded from:
`http://staff.scem.westernsydney.edu.au/~dave/bandana`
The BANDANA framework also includes the source code of an agent called ANACExampleNegotiator. Please take a look at this code to see an example of how to implement your agent. For an explanation of this code we refer to the general BANDANA manual.

We should remark that the BANDANA framework also includes classes to implement a full Diplomacy agent completely from scratch, but that is not what you are supposed to do. **Participants of the ANAC Diplomacy Challenge are not allowed to participate with a Diplomacy agent implemented completely by themselves. Instead, participants must only extend the ANACNegotiator class.**

This means that as a participant you cannot directly control which moves your player is going to make. You are only writing a negotiation algorithm. Your player will use an existing algorithm to determine which moves to make.

The `ANACNegotiator` is an abstract class that has a private field of type `ANACPlayer` which extends the `Player` class. Therefore, most of the options that you have to implement a player are hidden for participants of ANAC.

The `ANACNegotiator` class contains an abstract method with the signature `void negotiate(long deadline)`, which you must implement. This method is called at the beginning of each Spring phase and each Fall phase.

The given deadline is a Unix timestamp and it is your own responsibility to make sure that the method returns before this deadline. Your player must also implement the following main method and constructor:

```
public static void main(String[] args){
    MyANACNegotiator myNegotiator = new MyANACNegotiator(args);
    myNegotiator.run();
}

public MyANACNegotiator(String[] args) {
    super(args);

    //here you can put whatever you like...
}
```

You may give any name you like to your class, so you can change the name `MyANACNegotiator` to the name you like. Please make sure that it is a clear and recognizable name.

Once your agent has negotiated its agreements with the other players, the `ANACPlayer` will automatically choose a set of orders that obey those agreements and submit those. These orders come from the D-Brane Tactical Module as explained in Section 7 of the BANDANA 1.3 manual. You can use this same tactical module in your negotiation algorithm. You can access it by calling `getTacticalModule()`. It allows you to see which orders your player will submit if a certain proposal gets accepted. Based on this information you may decide to propose or not to propose a certain deal. Other tools that you may find useful for the development of your negotiator are the `Adjudicator` and the `GameBuilder`, which are also explained in the BANDANA manual.

As explained, you cannot directly control which orders your agent will submit, but you can do this indirectly, by making deals with the other players. For example, suppose you are playing as England and you want to invade Paris. In order to do so you could for example make an agreement with Germany that one of your fleets will move into Paris and that Germany will give you support. If Germany accepts this agreement, then your agent will indeed try to move your fleet into Paris. However, if your negotiation algorithm does not make such an agreement, then you cannot control whether your agent will attack Paris or not. In that case, if you want to find out whether your player will invade Paris or not, you need to call the D-Brane Tactical Module, and provide it with the agreements you did make (if any). It will then return the set of orders it will submit under those agreements.

Just like the `Player` class, the `ANACNegotiator` class has two public fields `me` and `game` which represent the power you play and the current state of the game. These are explained in the BANDANA manual. More information about the deals you can propose, how you can accept incoming proposals, and how the negotiation protocol works can be found in the BANDANA manual.

Finally, we should stress a couple of important differences between the things explained in the BANDANA manual, and the ANAC Diplomacy Challenge:

- Instead of extending the `Player` class you should extend the `ANACNegotiator` class.
- It is not possible to send informal messages (Section 6.8 of the manual) in the ANAC Diplomacy Challenge.
- You do not need to check whether previously confirmed deals are still valid (Section 6.3.3), because this is already done automatically by the `ANACNegotiator` class.
- You do not have to check whether deals are consistent with the confirmed deals (Section 6.7) because the Notary will do this.
- The BANDANA manual warns that you can never be sure the other players obey their deals. This is, however, not the case in the ANAC Diplomacy Challenge because each player will automatically obey all confirmed agreements.

6.1 Command Line Parameters

The constructor of your agent will call the constructor of its parent class `ANACNegotiator` with an array of Strings. The Strings in this array can be the same as the arguments of the other example agents provided with BANDANA. That is: you can set the name of the agent (-name), the log path (-log), the final year (-fy), the game server port (-gamePort) and the negotiation server port (-negoPort). All of these are only optional. If they are not specified then the `ANACNegotiator` will use default values. Any other strings passed to the super constructor will simply be ignored.

Please note, however, that in the ANAC Diplomacy Challenge the command line parameters will be set by the organizers, so you cannot set them yourself. Also, your agent should be able to start without setting any other command line parameters.

6.2 Reading from and Writing to Hard Disk

In order to debug your agent you may find it useful to write log files. You can do so by calling the getLogger() method from the `ANACNegotiatior` class and then call the logln() method of the `Logger` class (or any of its other logging methods). You don't need to call the enable() method because it will be enabled automatically.

It is important to know that your agent is only allowed to write to disk via this Logger class, and that **your agent is not allowed to read anything from hard disk**. Therefore, the log files are only useful while you are developing your agent, but cannot be used during the tournament itself (for example to store any knowledge about your opponents).

6.3 Bugs and Hacks

Of course the BANDANA framework may contain some bugs. If you find any such bug keep in mind that **you are not allowed to exploit such bugs to give your agent any kind of advantage**. Any of the classes or methods provided in the BANDANA framework may only be used in the way they were *intended* to be used, which should be clear from this manual or the BANDANA manual. Whenever you find a bug, or find any way to make use of a method or class that you are not sure of whether it is legal, please contact us.

7 How to Submit my Agent for the Challenge

Once you have finished implementing your agent you will have to compile it into a runnable jar file and put it in a zip file *together with your java source code*. You should send this zip file by e-mail to davedejonge@iiia.csic.es. Please make sure you put [ANAC Diplomacy] in the subject of the e-mail, and clearly include your name and the name of your agent in the e-mail. Also, you are required to provide a short description (100-300 words) of your negotiation algorithm when you submit your agent.

The deadline for submission is 21 May 2018 23:59 UTC-12.

When we receive your submission we will reply with a confirmation email. *If you did not receive a confirmation email, please contact us to make sure we have received your code correctly.*

By submitting your agent you are implicitly allowing us to add your agent to the BANDANA framework, so that it can be used by future researchers.

The winner of the challenge will be announced around 1 June. The winner (if there is any) is expected to have a representative attending the IJCAI 2018 conference to give a brief presentation describing his or her agent. IJCAI will take place from 13 to 19 July in Stockholm, Sweden.

8 Contact

If you still have any questions about Diplomacy, BANDANA, or the ANAC Diplomacy Challenge, please do not hesitate to contact us. Also if you have any suggestions on how to improve the BANDANA framework, the BANDANA manual, or this manual, then we are happy to hear from you. You can contact us by sending an e-mail to davedejonge@iiia.csic.es

References

[1] Dave de Jonge and Carles Sierra, *NB^3 a Multilateral Negotiation Algorithm for Large, Non-linear Agreement Spaces with Limited Time*, Autonomous Agents and Multi-Agent Systems 29 (5): pages 896-942, 2015