SillyNegoBot Architecture

Polberg Sylwia, Warsaw University of Technology

Supervised by Dr. Marcin Paprzycki

Table of Contents
Introduction........................................................................................................................................... 2
Reasoning behind architecture .............................................................................................................. 2
Architecture......................................................................................................................................... 6
  Hidden Layer Analysis.......................................................................................................................... 7
  Event Recording................................................................................................................................... 7
  Personality......................................................................................................................................... 8
  Agent modeling................................................................................................................................... 8
  Emotional System............................................................................................................................... 8
  Rational System................................................................................................................................. 9
  BDI ................................................................................................................................................... 10
Bibliography.......................................................................................................................................... 10

Note from the author: the SillyBot is a bot playing the Diplomacy game(Callhamer, 2000) at a 0
strategy level. It is created in the dip framework (Fabregues & Sierra, Dipgame). The SillyNegoBot is
its successor aiming for the level 1 (Fabregues & Sierra, Testbed for Multiagent Systems) and here we
will describe its model and purpose behind it. This document requires prior knowledge of the
language and idea of the Diplomacy game.
Introduction

Although automated negotiation itself is an interesting topic, it is already well researched and, unfortunately, use of automatic negotiations in a competition against humans does not perform well. Why is it so?

In this context let us know that humans do not always act rationally. They have emotions and prejudices that affect their decisions. Therefore, it is hard to predict their behavior as they often don’t choose the best possible solution. Moreover, humans have something called an intuition, and can decide to do something just because they ‘felt like it’ or had a ‘good feeling about it’. This is a challenge for an agent – how to successfully negotiate and win with such a mixture of logic and “chaos”. In the following paper an attempt at creating an agent architecture capable of such a thing will be presented.

Reasoning behind architecture

There is an old saying – ‘if you can’t beat them, join them’ - and it is the fundamental idea behind the SillyNegoBot model. In practice this basically means that to beat a human, bot will act like one. Let us first point out some basic characteristics of a negotiation bot:

- Analyzes current situation (both unit placement and relationships between powers)
- Is eager to negotiate or accept a deal, based on trust and possible benefits
- Models trust and reputation based on whether given agent keeps its part of a deal
- Chooses best possible plan

However, while it’s a good starting point – each of these elements needs improvement, and perhaps the list should be extended.

Get more data!

How to enhance board analysis? Its sole purpose is gathering data – the more we know, the better chances of winning we have. Therefore although for the level 1 it basic analysis might be sufficient, on the higher language layer we will have an option to ask for information and feelings. Moreover, it is not yet as human – like as it should. As a result our architecture should be open for adjustments and capable of handling situations such as when our bot says ‘I want to know whether Austria and Russia are allied’. This also creates a question – what if some data is needed, but no one wants to share it? There is a need for a system element that, thanks to already known data, can assume some facts with given certainty, for example ‘I am 50% sure Russia has proposed an alliance to Austria’. This probability is a very important notion and will be used also in other places – just as humans are ‘sure’, ‘almost sure’, or saying ‘this is rather unlikely’, a bot needs to be able to do that as well basing on any data he can get. This idea is based on plan probability used in negotiation bot developed by Angela Fabregues and Carles Sierra. Human players often come to situations when e.g. their units are
attacked by several powers, and this fact leads them to thinking ‘What if they are allied against me?’. A successful bot has to be able to simulate the ability to create some assumptions and work on them.

**Reasoning about data**

Let us now move a step further – we need to be able to create conclusions. For example, bot sees a German unit supporting Russian unit; what human sees is the possible alliance between Germany and Russia and the fact that they have showed it. In the first we expand the fact base and based on them we start to create conclusions. They can be separated into two levels; the primary level and the secondary level. Primary level creates conclusions from facts, while the other one derives conclusions from conclusions. For example, ‘Germany attacked Austria therefore they are at war’, ‘Italy attacked Germany therefore they are at war’ are from the primary level. Now imagine we know there is no attack history between Italy and Austria – we can say ‘Italy and Austria have never attacked each other hence they are neutral’. What we have is a situation of two neutral powers with common enemy. The natural way to do is out of the three conclusions mentioned above create another one – ‘Because Italy and Austria are neutral and both against Germany, alliance between them is highly possible’. This all resembles way humans think, and this is our aim.

Careful reader might have noticed the use of such notions like ‘attacked’, ‘allied with’, ‘being at war’. Of course given sentences will not be exactly like that in the SillyNegoBot, as building a human language parser is not aim of this project. However, we see that translating things like ‘MTO(BREAMY,PARAMY,FRANCE)’, ‘MTO(MARAMY,PIEAMY,FRANCE)’ (assuming that PARAMY in first and PIEAMY in latter are controlled by e.g. Germany) (Norman) can be just easily expressed as ‘France attacks Germany’. What we have obtained by such a translation is representing given fact in a human manner. People think about units moving, powers attacking, powers negotiating, not as sequences of MTO orders and so on. Moreover such expressions also carry some purpose and perhaps emotion. For example, MTO itself does not carry any information about message sender intention, while attack most definitely does. On top of that we are making the first step towards platform independence, as we are freeing our reasoning from protocol and syntax that were necessary for the game to move to computers.

To sum up, so far we have extended methods of situation analysis by translation, two level conclusion creation and assuming unknown messages. Is that enough?

**Emotions**

Of course not, we are still completely in the rational zone and even if now it resembles human thinking more than the basic model, it is still not “it”. What we lack are emotions, ability to assume how powers feel about something, to make emotional conclusions or be able to increase certainty of given conclusions. It is easy to notice that conclusion ‘Because Italy and Austria are neutral and both against Germany, alliance between them is highly possible’ is weaker than ‘Because Italy and Austria like each other and they are both against Germany, alliance between them is almost sure’. This is also very close to the second point of our list – eagerness to negotiate and accept a deal. A human can decide not to help an opponent, even though he can be trusted and deal is very beneficial simply because he doesn’t want to – for example, that particular opponent attacked him in the past, or
refused to help in some important moment, or has done anything that would cause emotional discomfort towards that opponent. Therefore what we need is to introduce and represent emotions in the system, and a way to analyze how given powers actions make other players feel. For example, in human terms if units of power A are getting closer to units of power B, power A might start feeling uneasy about it or cautious. Moreover, a player can for example ‘want’ something – ‘England likes Russia, so wants to help it’. Here, the most basic question is: what emotions should we take into consideration?

In this situation, higher language levels come in handy – even if SillyNegoBot due to time constraints will most likely stop at level 1, it should be extendable. Moreover, emotions are necessary in order for the bot to resemble human behavior now, not ‘later’ in the future. Higher levels give us a pointer about when the bot is meant to be able to express its emotions, not when it is supposed to start having them. Therefore choosing the emotion set that is used later in the future is a natural and reasonable decision. Level 5 language gives us following feelings: Very Happy, Happy, Sad and Angry. First thing that it is missing is just being neutral about something; however, it can be simply expressed as a lack of emotion and is not an important issue right now. Are those 4 (5?) emotions enough? Should we think about extending them?

In human terms we would take into account things such as ‘how much do I like/dislike that power?’, ‘do I like/dislike what given power just did?’, ‘do I feel like I can trust that power?’ and so on. Although ‘I like what he just did’ can be substituted with ‘I am happy with what he just did’, ‘I dislike this power’ it is not equivalent to saying ‘I am sad because of this power’. Being ‘sad’ does not necessarily stand opposite to ‘liking’. We can see that type of sympathy or its lack as an aggregated value based on opponents actions. Instead of constantly re-calculating our feelings about someone, we would just update this ‘like’ marker. Therefore emotion set does not need adding a new independent element but creating a variable based on existing ones. However, is it the same when we come to emotional trust? It cannot be expressed in terms of rational trust, where we only take into account whether someone is keeping his part of a deal or not. For instance, can we ‘like’ an opponent, but feel we should be cautious and same time not able to represent it with current emotions? Imagine a situation when an opponent we are fond of has been making us sad for several rounds, e.g. by making decisions we do not approve or being nasty to another power we like. This might make us like him less, but not necessarily by amount to stop negotiations. Moreover if after those several odd rounds his behavior comes back the way it was, a human can e.g. think ‘ok, that was just one-time weird thing’ and trust him back again, which can happen at a different rate then regaining ‘liking’. Based on this, we will want to create emotional trust, dependent on basic emotions. Therefore for time being there is no reason to increase the number of available feelings, but two derived factors – ‘liking’ and ‘emotional trust’ – are needed. They will go hand in hand with rational trust and reputation, not replace them.

**Personality**

Players differ – some cheat and lie, some don’t. Some forgive and forget, some hold grudges till the end of the game. There is also a concept of being greedy, when you can manipulate an opponent however you like just by promising him some land (e.g. by committing to move out of a province and
offering him to move his unit there). This all calls for creating a structure that will hold all such characteristics of a bot, its “personality”.

**Deciding what to do**

We come to the last point from the list - choosing the best possible solution. Having more data makes us judge plans more accurately, however, it is still not enough. As the IIIA-CSIC team has already stated – why should we attempt to do something that can give big benefits, but has success rate of only 1%? This is very like sacrificing a pawn in chess in order to hit the queen, or deciding not to participate in a lottery with low chance of winning and expensive tickets. Modification, taking into account the probability of success and size of consequences connected with failure is necessary. This success rate is also affected by what given person thinks his opponent is going to do, what calls for a plan analyzer. In its most basic form such analysis is even used by inexperienced players – ‘I will not move my unit closer to others because my enemy is next to it and I am sure he wants to control the province I occupy – losing a resource center is not worth it’. Furthermore, big impact on what someone will do what type of a person he is and what is his playing style; e.g. a player we find to believe in anything he is told is a different type of an opponent than a player that knows others will lie to him. Someone that focuses on defending his territory and tends to play ‘safe’ is also different and creates different movement probabilities than a person that is rather aggressive. This calls for an agent modeling – subsystem creating a profile of an opponent. Let us now transform what we have said into an architecture description.
Hidden Layer Analysis

Formally it should actually be the first thing processing the input from environment – this is a system element responsible for attempting to recover unknown messages exchanged between powers. This means it will create some facts with predefined probability, for example ‘Germany offered peace to Italy with probability 0.5’. However, due to time limits, it will be implemented as one of the last parts.

Event Recording

The need of storing history and processing it in the bot is rather obvious – we want be able to remember who sent what message, moved unit where etc. in order to be able to reason about it. We can also create a notion of ‘memory’ – which history elements and for how long should be kept. For example, a typical human player will not remember which unit moved where 10 rounds ago and he does not need that knowledge in order to win. However, he might want to remember, that 10 rounds ago some power cheated on another one. This might lead to assuming hatred of the victim towards the attacker. As we have stated in the previous section, we want to extend analysis of the game by what we name: translation and conclusions.

First of all, we go to the Fact Recorder, where messages are saved the way they are – e.g. ‘MTO(PARAMY,BURAMY,FRANCE)’. From here we head to the Translated Recorder, where we would get either ‘France moved its unit’ or ‘France attacked Germany’ (assuming that Burgundy belongs to Germany). Now it is time to make conclusions from that – the rational conclusion would e.g. be ‘France and Germany are at war’, while the emotional one ‘Germany is angry at France’ (we assume human players are not masochists). In order to hold such structures, a system of predicates will be created – for mentioned examples Moved(France) and Angry(Germany, France) could be the proposed forms.
**Personality**

Here we will store variables describing the ‘personality’ of a bot, thresholds after which we stop negotiation (e.g. because of dislike), how much weight does it put to emotions and how much to rational conclusions and so on. There is no point specifying all these points at this stage of the design, as this part will most likely undergo many expansions and changes during the implementation. Here we have some simple characteristics:

Boolean canLie;

List<int> likeWeight; - here we can define how weight is put to different types of actions or emotions; this is useful for expressing a person that ‘only looks at good sides’, that gains ‘liking’ fast when made Happy, but loses it slowly when made Sad or Angry and so on. Same applies for how ones opinion about a player that attacks him or supports him changes.

Int likeLimit; - we can define a value of liking below which we stop negotiation – ‘I hate you, I’m not talking to you’

**Agent modeling**

Basically this part is to be responsible for creating personalities for opponents – just like Hidden Layer Analysis it will be one of the last things implemented in the system. Although for level 1 it can be very simple, we can still for example observe whether given player is defensive or aggressive, or what type of deals he accepts more eagerly.

**Emotional System**

The diagram is rather unclear and will be corrected in the future, although the idea behind it is very simple. Emotions along with liking and trust will form so called Emotion Chart. Emotions themselves are stored as emotional conclusions first, and Liking and Emotional Trust as simple integer variables from 0-100 (like percents). They form an Emotion chart, and a set of them (one per power) is stored in the Relationships. Here is where Emotion Analysis comes; based on what we know we will create the second level conclusions of emotional type. This means that as an effect we want to receive a statement concerning feelings or desires, which in human terms would for example be ‘Germany is at war with Italy, therefore probably Germany doesn’t like Italy’, ‘Italy likes England and England likes Russia, so most likely Italy is neutral or likes Russia’ or ‘I hate Austria, I want to make it sad’. This system affects the Desires and Beliefs used in the BDI model.
The idea behind it is very simple; based on what we know, we want to create a plan to execute and analyze its risks that are highly affected by what opponents are going to do. Moreover here we will create the second layer rational conclusions, for example ‘Germany likes England and they are both against France, so Germany might ally with England’. This system affects the Beliefs and Intentions in the BDI model, and NOT the Desires, unlike emotional system – what we ‘want’ is not the same as what we ‘need’.

Rational System
BDI

What our subsystems have done so far was gathering data, analyzing it, creating conclusions and so on – in other words, increasing the knowledge base using which we want to decide on strategy of our negotiation. For agents there exist two main knowledge driven models – BDI (in form of Jadex in Java) and SOAR. Simply speaking we want to plug stuff in and get solution out. Although according to some opinions SOAR has more possibilities, we will use BDI model in this project – whether as Jadex when implementing, or only as a model will be decided when we come to implementing; unfortunately I am not yet able to state whether Jadex is flexible enough to handle the planned changes. Choice of BDI over SOAR caused by:

- SOAR is not implemented in the projects language – Java – and is very nasty to code in
- SOAR is not efficient enough to handle negotiation as fast as we would like to
- In SOAR we would have to model emotional and rational knowledge types; it is already there in BDI

Bibliography

