Modelling Rational Agents with a Genetic Algorithm

Matthia Sabatelli

August 1, 2016

Abstract

In the following sections we propose a computational simulation which aims to model the behaviour of some of the agents that could make use of the *spyGEM* system. The model combines knowledge of different research areas such as Epistemic Logic, Machine Learning and Multi-Agent Systems. Two agents have been modelled, a possible Sender and a relative Criminal. The goal of the first one is to encrypt and send a top-secret message as safely as possible, while the criminal's goal is to avoid this possible scenario. The agents have been modelled according to the *BDI Architecture*, a formal model that defines how rational agents act and behave (1), which has been modified and expanded specifically for this simulation. The individual parameters of this architecture have been estimated through the use of a genetic algorithm, this has been done in order to understand which parts of the architecture turn out to be more significant in a possible *spyGEM* scenario.

1 Introduction

Once the *spyGEM* project started, before moving our attention to a broader perspective involving big data storage, we considered our idea as a possible way of sending sensitive information in a very secure way. Thanks to the combination of the Advanced Encryption Standard algorithm and the biological security layers that make it hard to get to the point in which it is possible to sequence the DNA of a bacteria, *spyGEM* originally would have been the newest method to use for sending highly reserved information. Most of the research that has been done in the lab in order to make our system as safe as possible has been inspired by imagining possible scenarios involving the potential users of *spyGEM*. What might happen if the agents wouldn't have enough knowledge to successfully make *spyGEM* work? or what could happen if one agent would be in possession of enough sensitive information in order to make the whole procedure fail? These are only two out of the multiple questions that made it clear that in order to create an efficient product, there was need to understand and imagine future scenarios with the possible users of the system as main actors. The two main agents that turned out to be the most significant ones were a potential *Sender* agent that has as main goal hiding and sending as safely as possible a top-secret message and a potential *Criminal*, who wants to avoid this scenario. Imagining all the possible interactions between these two actors is of course impossible to achieve, however it is possible to simulate part of their behaviours assuming they are rational. In order to do so we have been inspired by work (1) where an Artificial Intelligence architecture based on Epistemic Logic is proposed for modelling rational agents. This gave us the opportunity to simulate different type of Agents and through the use of a genetic algorithm explore the different features that characterize this rationality. By exploring and computing the optimal features we have been able to identify the core points that have to be considered when building *spyGEM* in the most efficient way as possible. The following sections explain how the computational model has been built and trained in order to identify the most important aspects of the A.I. architecture, and how this knowledge has guided the whole lab procedure.

2 Architecture of the Model

The Agents have been modelled according to the *BDI Architecture*, an epistemic logic based system proposed in (1) in 1991. The paper states how important Beliefs, Desires and Intentions are in determining the behaviour of rational agents, more in detail the authors assert how every rational action taken from an agent is basically based on these three components. The combination of these features have an impact on the agent's future, in fact they have the ability to produce different actions and as a consequence give rise to different future scenarios. These possible scenarios, called *Worlds*, are modelled according to a temporal structure and are defined as *Time Trees*, every moment in the time tree is the result of an action based on the three previously mentioned BDI features and is defined as *Situation*. One Agent can move between different time situations until it achieves its goal. The three BDI components have a different impact on the agent's behaviour and are defined as follows:

- Beliefs: are defined as those worlds that the agent believes to be possible, technically every *Belief* is considered as a complete and independent *Time Tree* since there are no restrictions that can avoid an Agent to believe something. This feature is formalized as follows: $\forall$ w $\in$ W.
  Where w corresponds to a single time tree and W as the entire set of beliefs.

- Desires: are those worlds that correspond to the optimal future scenario an agent wants to achieve, an agent can be happy with multiple outcomes but usually the main *Desire* is a single one and can be expressed by the following formalism: $\exists!$ w $\in$ W

- Intentions: are considered as sets of intention-accessible worlds. These worlds are ones that the agent has committed to attempt to realize. Differently from the *Beliefs*, where all possible worlds are considered, the *Intention* feature is characterized by a strong component of realism since the worlds that are part of this feature are only the ones that the Agent thinks can be a possible *Desire*.

In addition to the three BDI features some extra parameters have been added in order to model the agents. The reason this has been done is related to the genetic algorithm that has been used in order to simulate their behaviour

and is explained into more detail in the next section. The parameters that have been added to the architecture are different according to the type of agent that has been modelled, however the general idea is the same one for both of them. In order to represent a potential level of *Beliefs, Desires* and *Intentions* in a *spyGEM* scenario some extra features related to this specific world have been added that might have an impact on all three BDI components and as a consequence on the agents' actions.
Regarding a potential *Sender* the three additional features that have been added are the following:

- γ: which corresponds to a particular level of *riskiness* the agent will take into account in his actions. The impact of this feature is different on the three BDI components, in fact a high level of riskiness makes the agent have a smaller set of total *Beliefs* since there is no reason to think about all those worlds that involve highly safe scenarios. On the other side it is possible that a high level of γ leads to the optimal desired world and as a consequence to a world the agent thinks requires commitment.
- ψ: is a variable that is related to the level of *efficiency* that the agent wants to pursue in its actions. In this case the higher this value is the higher the value of the three BDI components will be.
- α: corresponds to the *time resources* the agent is willing to invest when creating its BDI features, the impact of this variable is similar to the previous one. The more time that is invested by an agent, the higher the set of Beliefs will be and the overall commitment to the possible worlds towards the optimal scenario.

Regarding the *Criminal* Agent the meaning of ψ and α has been changed. The first value corresponds to the amount of knowledge the agent has about the different *Time Trees* while the second one is related to the amount of motivation and effort he is willing to put while interfering with the *Sender's* plan.

The idea of the model was to identify which of the three BDI components is considered as more important by a potential *Sender* agent who wants to hide and send an encrypted message as safely as possible and by a *Criminal* who wants to avoid this scenario. In order to find this out a genetic algorithm has been used on different randomly generated agents and the level of the three BDI features has been measured once the algorithm converged to the optimal result. Genetic algorithms are a very common machine learning technique that is used in different Artificial Intelligence optimization problems (2, 3, 4), they are largely inspired by natural evolution and work as follows: a random set of possible solutions is created, each of these solutions goes through a fitness function which measures how close these solutions perform to the optimal result that wants to be achieved. It is important to notice that the individual values that make it possible to converge to the optimal result are not known beforehand, the only information that is known is the final outcome which can be achieved in multiple ways.
Once the agents are evaluated by the fitness function a truncation process occurs, similarly as what happens in nature where only the fittest animals survive, also in the simulation the only agents that are allowed to be part and give rise to

a new generation are the ones with higher fitness scores. How new generations are created more into detail is presented in the next subsection.

2.1 The Sender

An Agent is created as a string of binary bits and is represented as a chromosome, each part of it represents the three BDI features together with the three additional parameters mentioned in the previous section. An example of a randomly generated agent is the following:

$$Chromosome_1 = \underbrace{010010011010}_{Beliefs}\underbrace{011101110101}_{Intentions}\underbrace{010100101010}_{Desires} \tag{1}$$

$$Beliefs = \underbrace{0100}_{\gamma}\underbrace{1001}_{\psi}\underbrace{1010}_{\alpha} \tag{2}$$

Every binary pair corresponds to a decimal number that if converted properly and summed together establishes the total fitness value of the chromosome. For the sender Agent an optimal value of 60 has been set, this turned out in the following fitness function, where n is the total length of the chromosome and the main aim is to find the optimal set of bits that satisfy it:

$$f(x) = \sum_{i=0}^{n} x[i] = 60 \tag{3}$$

For our *Sender* simulation the following parameters turned out to be the most suitable ones while considering optimal convergence performance and training time:

- 20 Simulations: a total number of twenty different simulations have been run in order to check if the results where coherent between each other.
- 20 Pools of 100 Chromosomes: the simulation starts with 20 randomly generated chromosome pools with 100 agents each. This is the first set of agents that is evaluated by the fitness function and according to the fitness scores the breeding process starts.
- Truncation Method: is the approach that has been used in order to keep the fittest agents allowed to reproduce themselves, in order to avoid overfitting issues the top 70% of the agents has been kept while the rest 30% has been discarded.
- One point single crossover: is the breeding method that has been used. Once two chromosomes are chosen as parents they give birth to two new children by exchanging 3 of their bits. This technique is shown hereafter:

$$Parent_1 = \quad \underbrace{\mathbf{010}}_{CrossoverBits} \quad 010011010011101110101010100101010 \tag{4}$$

$$Parent_2 = \quad \underbrace{\mathbf{011}}_{CrossoverBits} \quad 010000010011101111001010111101000 \tag{5}$$

$$Child_1 = \underbrace{010}_{Parent1Bits} \quad \underbrace{01000000100111011111001010111101000}_{Parent2Bits} \tag{6}$$

$$Child_2 = \underbrace{011}_{Parent2Bits} \quad \underbrace{01001101001110111010101010100101010}_{Parent2Bits} \tag{7}$$

- 1% mutation rate: some randomness has to be added to the newest generation in order to make the pool more varied and avoid local minima, in our case a 1% mutation rate has been set which means that one bit of one chromosome is randomly changed from 0 to 1 or vice-versa.

Figures 1 ad 2 show the performance of the genetic algorithm and the analysis of the three different BDI features. These results are explained more into detail in the Results section.

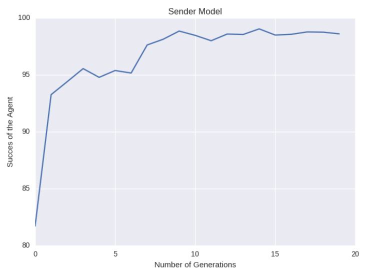

Figure 1: The graph shows how after 20 generations of breeding the pool of chromosomes representing the *Sender* agents converge to the optimal result. Generation 4 and 11 show some local minima that thanks to the random mutation present in the genetic algorithm don't impede the achievement of the optimal convergence.

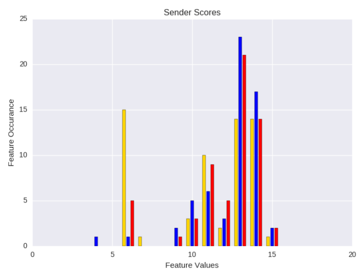

Figure 2: Histogram with the analysis of the BDI features. On the *x-axis* the decimal value represting that feature is presented while on the *y-axis* its amount of occurances during the simulation. The different colors in the historgram represent the different BDI components: Beliefs are shown in yellow, Desires in blue and Intentions in red.

2.2 The Criminal

As already introduced previously some of the additional BDI features have been changed while modelling the *Criminal*, however this doesn't have any influence on how the Agent is represented which still corresponds to a string of binary bits. In order to make the computational simulation more varied some changes have been made to the genetic algorithm starting from the fitness function which in this case has as optimal result a value of 65 as shown by the following equation:

$$f(x) = \sum_{i=0}^{n} x[i] = 65 \tag{8}$$

Also in this case while training the Agents 20 different simulations have been ran that started with 20 pools of 100 chromosomes each, however the truncation method kept the top 80% of the Agents and not 70% as in the previous case and the breeding technique that has been used was the n double crossover method. It consists in exchanging not only one single set of bits but two of them of 3 bits each as explained hereafter, the mutation rate has been kept to 1%.

$$Parent_1 = \underbrace{\mathbf{010}}_{CrossoverBits_1} \quad 01001101001110111010101010100101 \quad \underbrace{\mathbf{010}}_{CrossoverBits_2} \tag{9}$$

$$Parent_2 = \underbrace{\mathbf{011}}_{CrossoverBits} \quad 01000000100111011111001010111101 \quad \underbrace{\mathbf{000}}_{CrossoverBits_2} \tag{10}$$

$$Child_1 = \underbrace{010}_{Parent1Bits} \quad \underbrace{0100000010011101111001010111101}_{Parent2Bits} \quad \underbrace{010}_{Parent1Bits} \tag{11}$$

$$Child_2 = \underbrace{011}_{Parent2Bits} \quad \underbrace{01001101001110111010101010100101}_{Parent1Bits} \quad \underbrace{000}_{Parent2Bits} \tag{12}$$

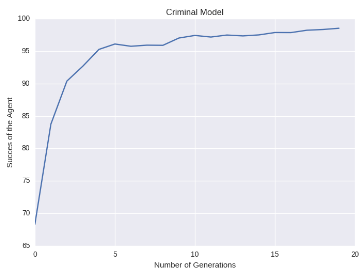

Figure 3: The performance of the genetic algorithm. Also in this case an optimal convergence has been achieved after 20 generations of training.

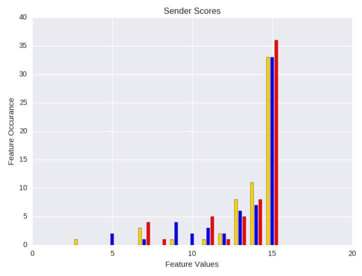

Figure 4: The analysis of the BDI features, the meaning of the graph corresponds to the one presented in Figure 2 however some significant differences in the histogram lead to the interesting conclusions presented in the next section.

3 Discussion and Conclusion

The aim of this section is twofold, on the one side it proves how it is possible to model artificial agents through the use of a genetic algorithm while on the other side it explains the relevance of this multi-agent simulation in the *spyGEM* project and how the relative results have been inspiring for the lab work.
Considering the first goal, as explained by Figures 1 and 3 it is indeed possible to modify the structure of the Agents in order to satisfy their relative fitness functions through the use of bio-computing techniques. The structure of the string of bits representing the *Sender* and the *Criminal* makes it possible to modify the Agents in a relatively short training time by converging to the optimal result already in 20 generations. These results show how this particular machine learning technique can be used for modelling Multi-Agent Systems and are in line with work (3). However the most remarkable and useful results have

been provided by the final analysis of the different BDI features representing the agents. The histograms show in fact a significant difference that has lead the development of the whole *spyGEM* project.
Considering Figure 1, the results show how in the case of the *Sender* model, the Belief feature is the one that the generations of artificial agents discard in order to get to the optimal score. In fact for 15 times as shown by the yellow histogram it is the BDI component with the lowest value, 6. This is remarkable since it shows how not too many resources should be invested in imagining all possible scenarios when sending a secret message, but it is better to have a strong set of Desires that nicely match with the Intention feature and pursue a single one of those. This can nicely be seen in the graph where an optimal value of 13 has been chosen by the agents for the Desire and Intention features for more then 20 times. The same is not true for the *Criminal* model where in order to be successful the generations of agents have to consider as equally important all the three BDI components. This is supported by the histograms that show how Beliefs, Desires and Intentions occur almost equally with a value of 15.
Considering this simulation and the main goal of *spyGEM* that sees a potential *Sender* communicating in the safest way as possible a message to a *Receiver* and a relative *Criminal* with an opposite goal we deduced that in order to make life as hard as possible to the *Criminal* we had to built the system in such a way that he has to consider as many *Worlds* as possible to be successful. This lead us to the idea of adding besides the A.E.S. method, when hiding the message into the DNA, a list of biological inspired security layers that have to be considered if the DNA has to be sequenced properly. As shown by the computational simulation, the knowledge of these *Worlds* has to be known by the *Criminal* if he wants to be successful since they correspond to the *Belief* feature.
Finally, it is important to mention that the computational model doesn't allow the agents to take real actions and maybe compete between each other, however it still provides important insights considering the features that make one agent as effective as possible assuming it is rational as defined by the BDI architecture.

4 References

1. Rao, A. S., & George, M. P. (1991). Modeling Rational Agents within a BDI-Architecture.

2. Herrera, F., & Verdegay, J. L. (1996). Genetic algorithms and soft computing. Physica-Verlag.

3. Zhong, W., Liu, J., Xue, M., & Jiao, L. (2004). A multiagent genetic algorithm for global numerical optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 34(2), 1128-1141.

4. Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. Machine learning, 3(2), 95-99.

It is possible to download the software that models the agents with the genetic algorithm from here: `https://github.com/paintception/myGEM`