Bayesian Networks for Micromanagement Decision Imitation in the RTS Game Starcraft

Ricardo Parra and Leonardo Garrido

Tecnológico de Monterrey, Campus Monterrey
Ave. Eugenio Garza Sada 2501. Monterrey, México
{a00619071,leonardo.garrido}@itesm.com

Abstract. Real time strategy (RTS) games provide various research areas for Artificial Intelligence. One of these areas involves the management of either individual or small group of units, called micromanagement. This research provides an approach that implements an imitation of the player's decisions as a mean for micromanagement combat in the RTS game Starcraft. A bayesian network is generated to fit the decisions taken by a player and then trained with information gather from the player's combat micromanagement. Then, this network is implemented on the game in order to enhance the performance of the game's built-in Artificial Intelligence module. Moreover, as the increase in performance is directly related to the player's game, it enriches the player's gaming experience. The results obtained proved that imitation through the implementation of bayesian networks can be achieved. Consequently, this provided an increase in the performance compared to the one presented by the game's built-in AI module.

Keywords: Bayesian Networks, RTS Video Games, Intelligent Autonomous Agents.

1 Introduction

The video game industry has been in constant development along the latest years. Numerous advancements have been made on graphics improvement in order to produce more realistic game environments, as well as the hardware that is required to handle these upgrades. The research area known as Artificial Intelligence (AI) has also its important place in the video game industry, among developers and researchers. AI is applied, on most part, to those units or characters that are known as non-playable characters (NPCs) in order to generate behaviours and as a medium for making decision, either autonomously or collectively. NPCs, as its name express, are all of the units on every game that are not controlled or modified by any player. A variety of algorithms, learning methods and reactive methods have been applied in games to provide a better gaming experience to the players.

Real time strategy (RTS) games are a stochastic, dynamic, and partially observable genre of video games [2]. Due to their strategic nature, RTS games require to continuously update the decisions, either from a player or the game's AI module. These type of games require throughout planning of strategies according to available information they have access to, such as disposable resources, available units and possible actions. The RTS game that was used along this research is titled "Starcraft: Broodwar"[3].

In the RTS game Starcraft, each player can posses up to a population equivalent to 200 units. Hence, the management of units can be divided into two segments: macromanagement and micromanagement. Macromanagement is usually regarded as the highest level of planning. It involves the management of resources and units productions. Micromanagement, on the other hand, refers to the individual control of units on combat. Micromanagement applied to a player's game require a large amount of time and precision.

Performing the necessary micromanagement tactics on 200, 100 or even 50 units can become a challenge that many players can't overcome. Hence, a method or process is required in order to continuously generate micromanagement decisions along the game or combat. Nevertheless, given that the environment provided by a RTS game can be partially observable, an approach that can cope with uncertainty is required. Moreover, if the output of the process can be match to an individual's personal strategy, the performance on a full scale game could be higher.

Considering the environment is partially observable, a bayesian network approach was implemented. Bayesian networks (BN) are one of the most appealed techniques to learn and solve problems while coping with uncertainty. BNs use probability and likelihood that specific success of events given a set of evidence. Moreover, it represents its model through compact acyclic directed graphs (DAG).

This paper exploits two vital tools that provides both graphical model and coded libraries, GeNIe and SMILE respectively, developed by the Decision Systems Laboratory of the University of Pittsburgh [4]. The GeNIe (Graphical Network Interface) software is the graphical model for the portable Bayesian network inference engine SMILE (Structural Modelling, Inference, and Learning Engine). GeNIe provides a graphical editor where the user can create a network, modify nodes properties, establish the conditional dependencies and much more. SMILE are a platform independent C++ libraries with classes supporting object oriented programming.

2 Related Work

In the latest years, video games have been exploited as research platforms for AI. The implementation of bayesian networks on games, as well as different approaches to micromanagement have capture the attention of different researchers. In Cummings et al.[5], a bayesian network approach for decision making was implemented in order to determine which chest to open out of two possible options. They use information recorded about previous open chests in order to calculate the probabilities for the CPT's of the nodes.

There has been previous work where bayesian networks have been implemented in RTS games. In their work, Synnaeve and Bessiere[6]worked under the assumption that decisions made by human players might have a mental probability model about opponents opening strategies. They used a data set presented on Weber and Mateas[7] work, containing 9316 Starcraft game logs and applied it to a bayesian model they proposed for opening recognition. They implement a backtracked model that helped them cope with the noise produced from missing data. The results involved a positive performance, even in the presence of noise due to removal of observations.

Further work of Synnaeve and Bessiere [8] involves unit control for the RTS game Starcraft. They propose a sensory motor bayesian framework for deciding the course of actions and the direction it should take. They make use of variables such as possible directions, objective direction and include other variables that represent damage, unit's size and priorities. They also applied potential fields that influence the units depending on their type and goal for complementing the decision making.

Moreover, more research related to micromanagement has been made in the work presented by Szczepanski and Aamodt[9]. They applied case based reasoning to the RTS game Warcraft3 for unit control. The case based reasoning received the game's state in order to retrieve the decisions of the most similar case and adapt it. Then, the unit receives a command with the information on how to respond to that specific case. This process is repeated every second during the experiment. The results from their experiments describe a consistent increase on the performance displayed by the AI against its opponents.

3 Bayesian Networks

The Bayesian Networks (BN), also referred as belief networks, are directed graph models used to represent a specific domain along with its variables and their mutual relationships. They make use of the Bayes' Theorem in order to calculate the probability of a specific event given known information, usually referred to as evidence. Bayes' Theorem is expressed as:

$$P(E|H) = \frac{P(H|E)P(H)}{P(E)} \tag{1}$$

BN are widely used in order to cope with uncertainty at a reasoning process at discrete sample space. BN use inference based on the knowledge of evidence and the conditional relationship between its nodes. Their implementation is included in decision theory for risk analysis and prediction in decision making. These networks make use of the bayesian probability, which was described by Heckerman [10] as a person's degree of belief on a specific event.

The bayesian networks contain nodes $(X_i..X_j..X_n)$ in order to represent the set of variables that are considered to influence the domain. The values they hold might either be discrete or continuous. Nodes are wired together by a series of direct connections represented by links or arcs. These arcs represent the relationship of the dependencies between variables. Hence, this networks are called directed acyclic graphs (DAGS). Further information regarding bayesian networks is presented by Korb [1] and Russell [2].

4 Bayesian Networks for Decision Imitation

A modeling process was established in order to imitate the decisions taken by the player. This process was applied to the experimental scenario described later on the paper. The process can be broken down into three different segments: Information extraction from the player, bayesian network model creation, and bayesian network prediction. Figure 1 illustrates the steps followed during experimentation.

The first step involves information extraction from player. A human player is briefed about the experiment. The human then plays on specific map for n number of cycles, where n is equal to 30. Meanwhile, the environment is sensed and a data log with the state of relevant variables of the environment is generated. There are a total amount of n logs generated. After the cycles are done, the logs are merged together to form a database, which will be used for the training of the bayesian network.

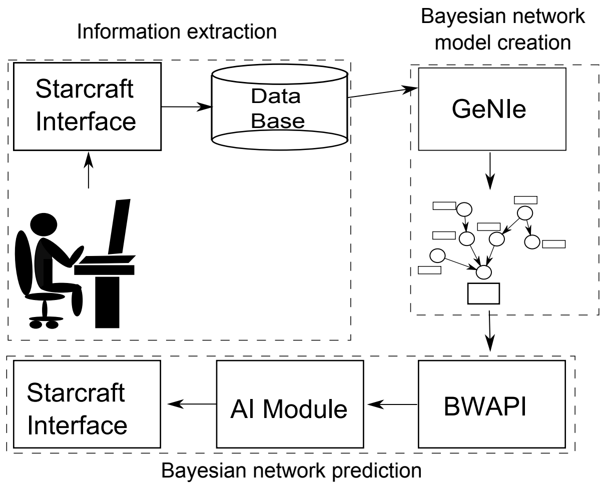

Fig. 1. Implementation Overview

The next step is the bayesian network model creation. The model is generated with the software known as GeNIe. Initially, we generate the nodes that represent the variables from the database that will be required for the decision making process. In order to select those nodes we queried an expert, the player whose decisions are being imitated, about the variables that influence his decision.

Then, the relationship between variables is sought and defined through arcs on the model. After the model is defined, the conditional probability tables are then filled with the calculated probabilities. These probabilities are obtained by loading the data extracted from the player and performing GeNIe's built-in parameter learning.

The last step involves the implementation of the bayesian network prediction. The experimental map is loaded once more with an modified Starcraft AI module that make use of the bayesian network. The environment is sensed by the AI module and the obtained information from the variables are used as evidence on the bayesian network. Once the propagation is made, we choose the action, direction, and distance with the highest posteriori probability as the one to be implemented. The sensing process and execution process are both made every five game frames. Given the game is running at the fastest speed, these processes are done, approximately, every one fifth of a second.

5 Experimental Setup

The Broodwar Application Programming Interface (BWAPI)[11] was used in order to implement the bayesian networks in the game. It was through it, as well, that a link between the game and the bayesian network's library was created. BWAPI generates a dynamic library (DLL) that can be loaded into Starcraft: Broodwar and enable a modified Starcraft AI module to be used.

The map used for the experiments was created on the Starcraft: Campaign Editor. The fighting area is composed of a 13 x 13 tiles diamond shaped arena. The fog of war, which forces the environment to be partially observable, was enabled. This limit the information available by the player's unit to that available on their field of vision. This property is also kept when the Starcraft AI module that contains bayesian network is used.

The units controlled by the player are situated in the middle of the arena while the opponent enemy forces are situated on the bottom part of it. Once the map is loaded, there is a 5 second time window for the player to select his units and establish them as hotkeys for better performance. After the time is spent, the enemy forces attack the player's units. The game ends when either forces is left without units.

5.1 Scenario: 2 vs. 3

In the scenario, we deployed two Dragoons from Protoss race and three Hydralisks from the Zerg race. The two Dragoons deployed by the player can defeat the three opponent's Hydraliks without any other need than a micromanagement well done. If it is not performed correctly, the game will end in the player been defeated.

The data was extracted from the interaction of the player with the setup previously mentioned on the foretold map. The information is extracted from the game every five frames throughout the player's game. The variables obtained for this scenario are declared on Table 1. The resulting database used for the training contains data from thirty repetitions. This scenario's training database contains over 8000 instances.

Table 1. Variables for 2 vs 3 Scenario

Friend1_HP	Next_Action
MyHP	
Attacked	
CurrentTargetDistance	
Next_Action	NextTargetDirection
Friend1_Direction	
Enemy1_Direction	
Enemy2_Direction	
Enemy2_Distance	NextTargetDistance
Enemy1_Distance	

We tried to keep the bayesian network as simple as it could without compromising the performance. Hence, we consulted the expert in order to design the corresponding network. The resulting network generated with the player's aid is illustrated in Figure 2. According to the player, the unit should decide his next action depending on his hitpoints, his ally hitpoints, whether it is targeted by an enemy or not, and the distance to his current target. The direction of the action to be done by the controlled unit depends on the next action as well as the direction of his ally and his enemies. In the scenario, a unit must consider the direction of his ally so they do not collied when they move to a safer place. Finally, the distance at which the action would be made is defined by the distance toward the enemies. Regardless the presence of a third enemy, the decision can be made by considering two enemies. Hence, it can be observed on the proposed network the lack of the third enemy.

6 Results

The results obtained from the scenario are presented in this section. First, we present the performance contrast between the Starcraft built-in AI module and the AI module containing the bayesian network. Then, a comparison between the decisions taken by the player and the decisions taken by the bayesian network AI module is made. In this comparison, a set of variables are selected and set to a specific discretized value. Moreover, the possible output corresponding to those values is graphed and compared.

The table declares the probability of choosing a specific next action, first column, given the available evidence, first two rows. In can be observed on the presented tables that the action chosen by the AI module resembles the decision taken by the player. This process was repeated for a series of other configurations of available information. The results are presented on Table 6.

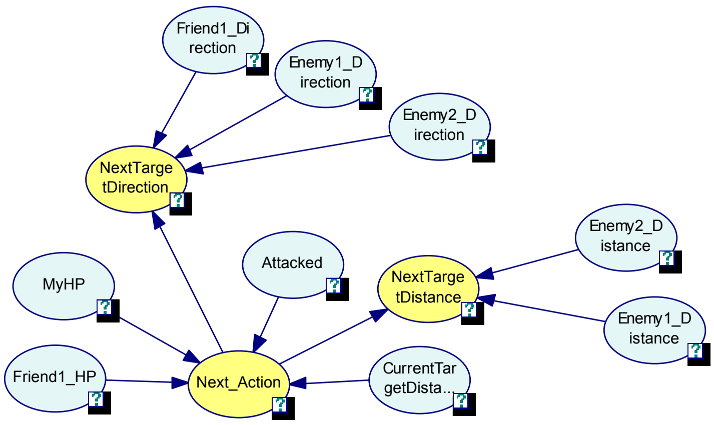

Fig. 2. Bayesian network model for a two Dragoons vs three Hydralisks

6.1 Scenario: 2 vs. 3

The first part of the results is a comparison between performances of the Starcraft built-in AI module and the module that implements the player's decisions. We elaborated 200 games for each AI module and obtained whether they achieved victory or not. There was a significant difference in performance between the AI modules. As express previously on the scenario's setup, the built-in AI module is not capable of defeating the opposing units. This caused that the built-in AI module generated 0% of victories. Nonetheless, if a bayesian network is used in order to imitate a player's micromanagement decisions, the expected percentage of victory increase to 44.5%.

The second part of analysing the results requires a comparison between the expected decisions according to the player's information and the decisions taken by the bayesian network AI module. We generated a series of tables that contains the probability distribution for a specific set of evidence on the game. We compared the course of action taken by the player, the training set data, with the one taken by the bayesian network AI module, the test set data. Both sets of data will be presented as tables representing specific environments and their prediction.

Table 2 contain the training distribution presented by the player, while Table 3 contain the distribution presented by the Starcraft AI module that implemented the bayesian network. The first column of the tables refer to the variable to be predicted and the corresponding states it contain. The states of the node are declared on rows on the first column. The rest of the columns in the tables express the combination the state or set of states established as evidence.

In Table 2 and Table 3 the variable to be predicted, labelled in the first column, is NextAction. The states of NextAction considered for the comparison are AttackMove, AttackUnit, PlayerGuard and Move. The rest of the columns establish the combination of specific data as evidence. Table 2 and Table 3 have

MyHP variable set as Medium, FriendHP set as Full, Attacked set as True and the distance to the current target,CurrentTargetDistance, with several possible values: Melee,Ranged1, Ranged2, and Ranged3. Therefore, all of tables presented declare the variable intended for prediction as well as the variables and states that are being established as evidence.

Table 2. Player's decisions over NextAction considering MyHP = Medium, Friend1_HP = Full, Attacked = True and CurrentTargetDistance

	Medium			
	Full			
	True			
Next Action	Melee	Ranged1	Ranged2	Ranged3
AttackUnit	66.67%	5.26%	0.00%	90.00%
Move	11.11%	84.21%	100%	5.00%
PlayerGuard	22.22%	5.26%	0.00%	5.00%
AttackMove	0.00%	5.26%	0.00%	0.00%

Table 3. Bayesian network AI module's decisions over NextAction considering MyHP = Medium, Friend1_HP = Full, Attacked = True and CurrentTargetDistance

	Medium			
	Full			
	True			
Next Action	Melee	Ranged1	Ranged2	Ranged3
AttackUnit	100%	0.00%	0.00%	97.80%
Move	0.00%	100%	100%	0.49%
PlayerGuard	0.00%	0.00%	0.00%	1.71%
AttackMove	0.00%	0.00%	0.00%	0.00%

Further comparison was made with the NextTargetDirection node. The results obtained on the experiments are encouraging. Table 4 presents the probability distribution of the training data in two different situations. Table 5 contains the probability distribution of the obtained performance of the modified Starcraft AI —module. Hence, it can be observed that the selection of the direction an action must be done resemble the selection observed on the player's data.

Finally, we present on Table 6 an overview of the behavior of correct and incorrect decision made by the bayesian network AI module. Correct decision refers to the match of the state of a variable with the highest probability between the player's decisions and the AI module decisions, such as the ones presented

Table 4. Player's decisions over NextTargetDirection considering NextAction = AttackUnit, Friend1_direction = Region5, Enemy2_direction, and Enemy1_direction

	AttackUnit	
	Region5	
	Region4	Region1
Next Target Direction	Region3	Region1
Region1	0.00%	92.68%
Region2	0.00%	3.41%
Region3	0.00%	1.46%
Region4	100%	0.00%
Region5	0.00%	0.98%
Region6	0.00%	1.46%

Table 5. Bayesian network AI module's decisions over NextTargetDirection considering NextAction = AttackUnit, Friend1_direction, Enemy2_direction = Region1, and Enemy1_direction = Region1

	AttackUnit	
	Region5	
	Region4	Region1
Next Target Direction	Region3	Region1
Region1	0.00%	96.80%
Region2	0.00%	0.00%
Region3	0.00%	2.36%
Region4	100%	0.00%
Region5	0.00%	0.00%
Region6	0.00%	0.84%

Table 6. Bayesian network AI module's decisions performance over 50 different situations

	Bayesian AI module
	Scenario 2 vs 3
Correct Decision	80%
Incorrect Decision	20%

on previous tables. Incorrect decision refer to the existence of discrepancy on the chosen state of an output between the player's decisions and the AI module decisions, given the same evidence is presented. An example of correct decision can be shown by considering Table 4 and Table 5. The correct decision refers to the match between tables where Region4 is selected by both, the player and the bayesian network AI module, given the first set of evidence.

6.2 Discussion

The proposed method was designed to imitate player's decisions in the RTS game Starcraft. By implementing belief networks we can make use of an expert's opinion, in this case a Starcraft player, in order to establish a bayesian network model that suits his decisions. Moreover, it is complemented by applying the knowledge of the player game information to obtain the conditional probabilities of the network. It can be observed on Table 6 that the correct imitation of decisions of the player's decisions done by the bayesian network AI module is done with a high accuracy rate.

The performance observed by the bayesian network AI module excels the performance obtained from the Starcraft built-in AI module. The 44.5% of victories provided by the experiments establishes the increase on it. This percentage is partially low given the attacks made by the default Starcraft AI module does not follow the same pattern every time. For example, the enemies may all attack the same unit controlled by the player, or they can split to attack both the player's units.

Hence, further scenarios were tested as well and their average performance exceeds the 60% of victories. It is clear that by introducing an external influence to the built-in Starcraft AI module an increase on performance can be made. Further research on imitating decisions can be made using the RTS game Starcraft as test-bed.

7 Conclusion

We presented a bayesian network approach for unit micromanagement in a RTS game. The results obtained in this research support the hypothesis of a performance improvement on the Starcraft built-in AI module. The importance of the increase 44.5% in victories is significant given the fact that the default performance is of 0% victories. Moreover, this method enables a performance that resembles that of the player. In a full RTS game, the advantage of having unit you controlled synchronized with you own strategies can enrich the gaming experience for the players. The results also support the fact that research on bayesian network might lead to interesting work on imitating decisions taken by humans. The bayesian networks provide a stable, understandable and transparent method to generate the decision imitation.

There is future work to be done in our research. The proposed learning method in our research is based on an offline learning. Further work can involve a dynamic updating on the belief network while the player is interacting with the game. This can enable online learning in order to train a bayesian network on a full game rather than on a specific map.

References

1. Korb, K., Nicholson, A.: Bayesian Artificial Intelligence. Chapman and Hall Editorial (2010)
2. Russell, S., Norvig, P.: Artificial Intelligence a Modern Approach, 3rd edn. Pearson Education (2009)
3. Blizzard Entertainment: Starcraft, `http://us.blizzard.com/en-us/games/sc/` (accessed in January 2012)
4. Decision System Laboratory, University of Pittsburgh: GeNIe & SMILE, `http://genie.sis.pitt.edu/` (accessed January 2012)
5. Cummings, J.: Bayesian networks in video games. Pennsylvania Associataion of Computer and Information Science Educators (2008)
6. Synnaeve, G., Bessiere, P.: A Bayesian Model for Opening Prediction in RTS Games with Application to StarCraft. In: IEEE Conference on Computational Intelligence and Games (2011)
7. Weber, B.G., Mateas, M.: A Data Mining Approach to Strategy Prediction. In: 2009 IEEE Symposium on Computational Intelligence and Games (2009)
8. Synnaeve, G., Bessiere, P.: A Bayesian Model for RTS units control applied to StarCraft. In: IEEE Conference on Computational Intelligence and Games (2011)
9. Szczepanski, T., Aamodt, A.: Case-based reasoning for improved micromanagement in Real-time strategy games (2008)
10. Heckerman, D.: A tutorial on learning with Bayesian Networks. Microsoft Research, Advanced Technology Devision: Microsoft Corporation, US (1995)
11. BWAPI - An API for intereacting with Starcraft: Broodwars (1.16.1), `http://code.google.com/p/bwapi/` (accessed in January 2012)