Using Bayesian Network in Plan Recognition for RTS Games

Danny C. Cheng  
College of Computer Studies  
De La Salle University, Manila  
0917-5291560  
chengd@dlsu.edu.ph

Romnel Louis L. Agcaoili  
College of Computer Studies  
De La Salle University, Manila  
0922-3455151  
iceblast_6@yahoo.com

Ramon Carlos O. Ang  
College of Computer Studies  
De La Salle University, Manila  
0917-7920032  
i_love_mamon@yahoo.com

Dennis Ivan O. Li  
College of Computer Studies  
De La Salle University, Manila  
0916-2407202  
sinnedsynd@yahoo.com

Carlson Erwin N. Tang  
College of Computer Studies  
De La Salle University, Manila  
0919-4100863  
cyrston@yahoo.com

ABSTRACT
Real-time strategy games such as Warcraft, which is very popular, still utilizes finite-state machines and rules to determine the artificial intelligence of the NPC or Non-player computer. This provided a stable implementation for the game but suffers from predictability of the computer players actions. In order to further improve the current state of technology and limit predictability of the NPCs artificial intelligence, user modeling and plan recognition is applied to allow for the NPC to learn and adopt to the human player. The UMAGA agent is implemented on top of the Stratagus game engine thus being similar to the actual Warcraft II game. It consists mostly of a user modeling agent that uses Bayesian Inference in plan recognition. It uses the information gathered from the user like the current state of a strategy game and previous actions to make a prediction of what the user would do next. For the current research, this model is used to determine actions that would assist the human player in the duration of the game play.

Keywords
User Modeling, Plan Recognition, Bayesian Networks, Games, Artificial Intelligence

1. INTRODUCTION
Bayesian Inference has been used in first-person shooter games like Quake 3 (bot decision making) and other applications unrelated to games like Email spam protection. This is because Bayesian Inference can make artificial intelligence work faster, as seen with the response times in Quake 3 bots, and more accurate, as seen in the little amount of spam that gets through in Email accounts. The research applies Bayesian Inference in Real-Time Strategy Games (RTS) in particular the Agimat game.

Agimat is similar to Warcraft II, a popular RTS game, in terms of gameplay elements like resource management, building of bases, and management of units. The differences lie on the graphics, and unit statistics. In this type of environment, a good way of helping users would be by assisting them with the different game elements mentioned above as these elements may overwhelm the user.

This assistance can be provided by adding a user modeling agent into the environment as user models are used to capture the behavior of the user and use this information to help the user.

UMAGA is a user modeling agent that assists in a real-time strategy game and is implemented in Agimat. It is able to capture the behavior of the user through the different actions performed by the player (i.e. attacking, building units) and the effects on the environment during the game or the state (i.e. gaining gold, killed units). The series of actions and states represent what a user is doing during a game under different scenarios and this is what comprises a strategy.

The behavior captured from the user is represented through behavior attributes. Behavior attributes are values that represent a user’s behavior in different aspects of a strategy game like gathering resources or amassing units. Similar behaviors are then clustered together. Clusters represent the strategies that the user has performed in similar game scenarios. The model would enable the agent to assist the user by predicting their would-be actions using the clusters and help them perform these actions.

Assistance can be achieved through plan recognition in particular, by using Bayesian Network. Bayesian Network was used because of the network like structure of the user model brought about by clustering of user strategies. Prediction of the user’s strategies involves the following steps: monitoring the current game, comparing the game state and behavior of the user to the most similar state and behavior in the user’s model, and performing the strategy this state entails for the user. The paper has three sections. The first section of the paper elaborates about the system that used Bayesian Inference for quake. The second part talks about Bayesian Inference used by the UMAGA agent.
Finally, the last part deals with the conclusions, problems encountered and recommendations.

2. RELATED WORK

Wargus is a real-time strategy game cloned from the popular game Warcraft II. Wargus is developed from an open source engine Stratagus, that is used to make RTS games. The choice of Wargus is greatly influenced by the availability of the code for manipulation thereby making experimentation easier.

Wargus covers all basic functions of a typical RTS game. It is focused on military command wherein the player has a "god's point-of-view" of the world. All information on the units and buildings of the player are available. Map and enemy information are hidden at the start of the game. The player has the choice of which unit and building type to built as well as which technology to research on, among others.

Implementing user modeling in Wargus would entail modification of the artificial intelligence (AI) part of the program code. Additional features should be added to allow for the presence of the modeling agent, as well as capability for knowledge acquisition and analysis by the said agent.

Some systems that are related to this research are the COMETS [1], a system that uses case-based plan recognition in predicting a user's actions using the game space invaders as its domain, and CAT [5], a system that uses case-based plan selection in trying to defeat its opponent using the Wargus game as its domain, systems.

The relevance of the COMETS system to this paper's research is the representation technique, which makes use of plans containing different states of the environment and the actions the player performed during the states in the environment, and its objective, to predict the user's actions.

Although the COMETS system provides a method for predicting a user's actions, the domain where it was used was a much simpler domain than this paper's research. The COMETS systems possessed only three possible states, the safe, unsafe, and very unsafe states. The environment where this paper's system will be implemented in is more complex than the COMETS system. This is where the CAT system comes in.

The significance of the CAT system is that system's opponents are dynamic and that the system uses the same domain as this paper's research, a real-time strategy game. Another significance of the CAT system is its usage of the Tielt, which is a tool that integrates AI systems with an environment. The Tielt tool is important to this paper's research because it gives a possible solution on integrating this paper's system with the environment.

Even if the CAT system domain is similar to this paper's domain, the objectives of the agents are different. In CAT it tries to defeat its opponents by selecting appropriate strategies to win. In this paper's system, it tries to predict the user's actions so that it may assist the user in executing the task.

All Plan recognition in computer games has recently drawn interest among researchers. Fagan and Cunningham [1] applied case-based plan recognition (CBPR) in a simple Space Invaders game. The system they developed was able to predict a player's behavior in real-time, albeit the game involved relatively few sets of states and actions.

Cheng and Thawonmas [2] proposed applying CBPR in a RTS game. This approach is more complicated given the characteristics of RTS games. Unlike Space Invaders, RTS games involve more game elements to consider, tasks to simultaneously manage, among others - all in real-time. CBPR in RTS game could then be used to assist the user in playing the game. This paper applies the same concept of assistance, using Bayesian inference and a user-specific approach in plan recognition.

Bayesian inference has also been applied in computer game research. Thurau et al. [3] used Bayesian imitation learning to produce “life-like computer game agents” in QUAKE II®. They use a Bayesian approach in selecting the next action for the computer game agent. The computer agent compares its past state, action, and other pertinent game information to those generated by human players and performs the same action performed by the human as recorded in the data. Their research shows that Bayesian models can be adopted on computer game agents that are expected to be dynamic and to display human-like behavior.

3. ACTIONS, STATES, AND BEHAVIOR FACTORS

The game states and actions performed by the user are essential in order to predict his plans. This pair serves as the user model for the NPC. The research considers thirteen game elements comprising the state vectors. The actions of the user are used to assign scores for him in terms of how he plays a RTS game. This section discusses the plan recognition function of the UMAGA agent using a Bayesian network.

3.1 Actions and Behavior Factors

Our agent uses Bayesian plan recognition to infer the next behavior that the user might display, not the next (singular) action the user will perform. To do this, actions are first obtained from the user's history. Adjacent actions under similar game states are then grouped into strings which are called trims. Trims contain state-action pairs which do not differ significantly in terms of state values. The action sequences contained therein are abstracted into behavior factors.

Behavior factors reflect the different aspects of a RTS game - economic, expansion, unit massing, security, structure assault, and unit assault. Actions within a particular trim are scored according to how they reflect a behavior factor. For instance, a string of actions consisting of training workers and assigning them to mine resources will produce a high score for the economic aspect and will not yield any score for unit assault. Each trim, therefore, has a score for each of the six behavior factors.

3.2 Game States for a RTS Game

States reflect the overall situation of the game environment. In this application, the state consists of thirteen game elements namely: resources, total razings, total kills, farm count, tower count, melee building count, range building count, magic building count, main building count, soldier count, archer count, mage count, and worker count. Like actions, the states are considered
collectively within a trim. Each trim contains a representative state to describe the environment within a given instant.

3.3 Clustering Similar States and Behavior Factors

Each element in a representative state, or any state vector for that matter, can be measured numerically. Given the nature of RTS games, it is possible, and very likely, that two states may be similar (e.g. high in resources, few workers, many soldiers, etc.) and yet differ in terms of the numerical data they contain. For instance, a state with resource value of 5000 and worker count of 10 may describe a practically similar situation with a state having a resource value of 5,500 and 11 workers although both differ numerically. The same notion applies for behavior factors. Two trims may reflect numerically different situations but indicate similar user behaviors (e.g. focused on economy and massing units, low priority in fortification/security, etc).

To identify similar states and behaviors, \( k \) means clustering is performed for the two components. Performing \( k \) means clustering for states involves thirteen (13) axes, each representing one of the thirteen game elements. Six (6) axes are required to cluster behavior clusters via \( k \) means, where each axis represents one of the behavior factors mentioned above. From then on, states and behaviors are identified according to the clusters they belong to. Behavior clusters are henceforth treated as the user’s plans. High scores for certain aspects indicate that the user prioritizes those aspects of the game. Low scores, in turn, indicate low priority by the user for the behavior factors involved. When viewed as a plan, these behaviors dictate the actions that the user will perform.

4. BAYESIAN NETWORK IN PLAN RECOGNITION

A Bayesian Network is used in plan recognition in order to infer future actions. This involves applying the Bayes rule to construct the network and use the values of the edges in order to predict the probability of events, as expressed by nodes, in occurring.

4.1 Baye’s Rule

The Baye’s Rule [4] is expressed as:

\[
P(A|B) = \frac{P(B|A)P(A)}{P(B)}
\]

where:

- \( P(A) \) is referred to as the prior probability of \( A \),
- \( P(B) \) is called the marginal probability of \( B \),
- \( P(B|A) \) is called the conditional probability of \( B \) occurring given that hypothesis \( A \) is true,

and

- \( P(A|B) \) is referred to as the posterior probability of \( A \) given \( B \); the probability that \( A \) occurs given \( B \).

4.2 Constructing the Network

In this paper, the Bayes rule is applied using states and behaviors as parameters in order to construct the network. The Bayesian network consists of nodes that can either be a state or a behavior. Directed edges among the nodes indicate the probability of occurrence of the child node given the values of its parent nodes.

Current states and the player’s previous behavior serve as parent nodes to the user’s next behavior. Each behavior in the node can be considered as a parent node, if used as basis for predicting, or as a child node if used as a possible behavior. States are always considered as bases for predicting only. The next behavior of the player is predicted by referring to the current state of the game and the most recent behavior displayed by the user. Note that states and behaviors are recognized as clusters of similar elements and notated as:

\[
\text{next B: next behavior} \\
\text{previous B: previous behavior} \\
\text{current S: current state}
\]

The probability of a behavior is then predicted as:

\[
P(\text{next B} | \text{previous B} \land \text{current S}) = P(\text{next B} | \text{previous B}) \cdot P(\text{next B} | \text{current S})
\]

where:

\[
P(\text{next B} | \text{previous B}) = P(\text{previous B} | \text{next B}) \cdot P(\text{next B})
\]

\[
P(\text{previous B})
\]

\[
P(\text{current S} | \text{next B}) \cdot P(\text{next B})
\]

\[
P(\text{current S})
\]

Prediction of the user’s next behavior gives equal weights to the current state of the game and the player’s most recent behavior. The game state provides the context by which a player may behave during the game. For instance, having very low resources will most likely compel the user to focus on economy. The user’s previous behavior, likewise, may dictate future behaviors. Consider a scenario where the user has amassed a significant amount of soldiers, as indicated by the game state. Combining the game state and previous behaviors may show that a player focuses on massing attack units (Massing being one of the behavior factors), until a certain amount of soldiers is attained. After which, the player proceeds to attack the enemy base. The game state and previous behavior can thus lead to predicting the player’s next behavior.

The value of a state to a behavior is derived by obtaining the probability that a game state will lead to a certain behavior for the user. This is obtained by dividing the number of elements in a state cluster that point to the behavior cluster in question, over the number of elements in a behavior cluster. For example, a value of 0.67 from a state 1 to behavior 1 (see Figure 1) means that two-thirds of the elements within the state cluster led to the player performing a series of actions that was classified to be within the behavior cluster. The remaining one-third led to actions that did not belong to that behavior cluster. The same concept applied to values of edges among behaviors. The number of elements
belonging to the parent behavior that point to the child behavior is divided over the total number of elements in the parent behavior.

**Figure 1. State to Behavior Probabilities**

**Figure 2. Behavior A can be predicted given State A and Behavior B**

4.3 Plan Recognition Using Bayesian Network

In the context of this research, the player’s plan is expressed in terms of his behavior in the game. The behavior factors capture the different aspects of a RTS game that the player may or may not prioritize at the moment. The player’s focus on certain behavior factors shows his plan/behavior.

In order to recognize the player’s next behavior, the state and previous behavior displayed by the player have to be classified according to the state and behavior clusters that they belong to, respectively. The Bayesian network is then consulted in order to obtain the behavior cluster with the highest prediction probability - that is, the cluster with the highest value prediction value derived by applying the Bayes rule (see Section 4.3) to the state and behavior edges that lead to it.

Figure 3 illustrates a small-scale Bayesian network. Observe that it is possible for a behavior node to point to itself (behavior cluster C). This happens when the user exhibits the same behavior even across different game states. The preceding and succeeding behaviors stored in the trims therefore belong to the same cluster. An example of this scenario would be extensive focus in attacking the enemy structures for an extended period of time. Actions involved in this behavior would be attack commands issued to military troops. A player can opt to continuously attack the enemy even if the other RTS game aspects are directly or indirectly affected (e.g. running low in resources, increasing amount in kills/razings, diminishing number of soldiers, etc). Barring other factors, consecutive behavior scores would be classified under the same behavior cluster even if their corresponding states would be in different state clusters.

**Figure 3. A Bayesian Network of State and Behavior Nodes**
Table 1. Values of the states for each cluster

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Resources</th>
<th>Kills</th>
<th>Workers</th>
<th>Melee</th>
<th>Range</th>
<th>Magic</th>
<th>Farms</th>
<th>Melee Building</th>
<th>Range Building</th>
<th>Magic Buildings</th>
<th>Man</th>
<th>Tower</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.1</td>
<td>3.3</td>
<td>21.667</td>
<td>8</td>
<td>4</td>
<td>0</td>
<td>9.375</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>4.1</td>
<td>3.333</td>
<td>25</td>
<td>13</td>
<td>4</td>
<td>0</td>
<td>15.25</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>7.5</td>
</tr>
<tr>
<td>2</td>
<td>13.4</td>
<td>3.333</td>
<td>26.667</td>
<td>8</td>
<td>4</td>
<td>0</td>
<td>25</td>
<td>60</td>
<td>40</td>
<td>0</td>
<td>40</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3.333</td>
<td>26.667</td>
<td>28</td>
<td>20</td>
<td>0</td>
<td>37.5</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>5</td>
</tr>
</tbody>
</table>

5. INITIAL EXPERIMENTS AND RESULTS

In order to test the Bayesian network used for plan recognition four different strategies were performed. The strategies focused on base development, base fortification, massing units and gathering resources. The Base development strategy focuses on building Farms, Barracks, Gnomish Inventors, and Mage Towers. Base fortification on the other hand focuses on building towers, base structures and soldiers. Massing units focuses on gathering a lot of troops. Finally gathering resources just keeps building as many workers and farms as possible. The training of workers and harvesting of resources was done during the early part of all games. They were performed only up to the extent when the user has enough resources to build the appropriate structures. These four strategies were played in 20 games, four of which were manual and the other 16 games were just duplicates of the four manually played games. The data was duplicated to provide a simulated model that has undergone several games. The user model at the end of the 20 games is seen in Table 1.

The following describes the data in Table 1:

State Cluster 0: The game state has very little resources, few workers, few melee and range units, very few farms and all building creation types.

State Cluster 1: The game state has little resources, few workers, few melee and range few farms, few melee buildings and a few towers.

State Cluster 2: The game state has moderate resources, moderate workers, few melee and range and moderate farms, several melee buildings and moderate range buildings, moderate main buildings.

State Cluster 3: The game state has few resources, moderate workers, moderate melee and range, moderate farms and all building types.

Table 2. Values for the behavior of each cluster.

<table>
<thead>
<tr>
<th>Economy</th>
<th>Expansion</th>
<th>Security</th>
<th>Massing</th>
<th>Army Assault</th>
<th>Structure Assault</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>31.25</td>
<td>36.833</td>
<td>0</td>
<td>10.86</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>8.333</td>
<td>34.167</td>
<td>0</td>
<td>8.384</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>54</td>
<td>0</td>
<td>10.784</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
<td>49.667</td>
<td>0</td>
<td>9.524</td>
<td>0</td>
</tr>
</tbody>
</table>

The following describes the data in Table 2:

Behavior Cluster 0: User is moderately concentrating on gathering resources and expanding base, he is giving a little focus on gathering troops.

Behavior Cluster 1: User is moderately concentrating on expanding base and giving a little focus on gathering troops and resources.

Behavior Cluster 2: User is focusing a little on gathering troops.

Behavior Cluster 3: User is moderately focusing on gathering the base and giving a little focus on gathering troops.

After the user model was created, a game was played with a strategy of base expansion. This would involve using behavior 0 and state 0 both of which represent the early parts of a game which focuses on gathering resources and being expansion minded. After building two towers and one farm the agent predicted the user performing behavior cluster 0. According to the user model, the cluster with the highest possibility with a value of 25% is cluster 0 which still suggests base expansion. The possibilities are seen below:

Table 3: Prediction values for state and behavior cluster 0 based on the 20 games

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Prediction Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.250000</td>
</tr>
<tr>
<td>1</td>
<td>0.125000</td>
</tr>
<tr>
<td>2</td>
<td>0.000000</td>
</tr>
<tr>
<td>3</td>
<td>0.000000</td>
</tr>
</tbody>
</table>

Although the agent was able to predict behavior cluster 0 which was the intention of the strategy, it was not able to provide assistance anymore after that.

6. CONCLUSION

We have presented an application of Bayesian network to plan recognition in a RTS game. Although full testing has not yet been performed on the UMAGA agent regarding its accuracy and reliability, the initial tests show promising results of the implementation of Bayesian network for plan recognition.

The results show accurate prediction during the early game as seen in the base development game. However prediction in the middle to late game happens very seldom. The prediction value table was analyzed and it was found out that there are many entries that have a value of zero. The agent was designed to provide assistance only if there is a value greater than zero because it suggests that the user has already performed this
strategy. The abundance of zeros is attributed to the current Bayesian formula because some values are negated when a zero value is encountered. This suggests a little revision in the Bayesian formula which would then make it more effective. Averaging the values of the edges from the state to the behavior in question together with that of the previous behavior to the behavior in question may be a good start. This eliminates the possibility of yielding a zero value for prediction if at least one of the said edges has a zero value.

7. REFERENCES