Adaptation of an Adversarial Non-player Character through Case Based Reasoning

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

Gian Kristian A. Fontanilla  
College of Computer Studies  
De La Salle University, Manila  
0917-9038259  
i_am_ife@yahoo.com

Anne Marie M. Africa  
College of Computer Studies  
De La Salle University, Manila  
0916-3260150  
ayanamiyuy@yahoo.com

Karmela Angela G. Cortez  
College of Computer Studies  
De La Salle University, Manila  
0921-4929747  
melacortez@yahoo.com

Paul Michael O. Go  
College of Computer Studies  
De La Salle University, Manila  
0917-3205668  
paulmichaelgo@yahoo.com

KEYWORDS  
Case-based Reasoning, Real-time Strategy Game, online adaptation

ABSTRACT  
Game development is now turning to other innovations such as applying Artificial Intelligence (AI) techniques [4]. However, such algorithms only make use of simple decision-making and still lack the ability to learn [2] One type of game that requires this kind of learning is real time strategy games. This research intends to present the CAN system that is designed as an adversarial Non Player Characters (NPC) that learns strategies in a real time strategy (RTS) game using Case-based Reasoning. Using the strategies learned from the past actions of the human player, CAN is able to adapt to current situation and change strategy online.

1. INTRODUCTION  
Real-time Strategy games offer environmental characteristics which AI researchers find attractive: partially observable, complex, continuous, and dynamic. In this domain, the environment changes dynamically even without the user’s intervention. The game features several entities and objects that interact with each other at a very fast pace. Therefore, as the game progresses, the possibilities grow exponentially. [3]

Until now, artificial intelligence techniques are still mostly made up of heuristics techniques and finite state machines, since these are well understood in the context of computer gaming. Finite state machines provide agents very limited responses to certain input given a certain state, and are only feasible when there are a relatively small number of states in the game. In addition, finite state machines make agents [4]

predictable since they are finite and respond in the same manner to same inputs in the same state.

Real time strategy games, however, have a very large number of states, which also grows exponentially as the game progresses. It would be tedious to encode all possible actions for all possible states in a certain game. An alternative approach would be to have the AI learn to adapt to a state or situation in a game. Adapting to a certain situation must be dynamic since enumerating all possible states would be close to impossible. The paper talks about Case-Based Reasoning, an approach which enables an agent to adapt and learn from previously encountered situations, applied in an adversarial Real-Time Strategy game opponent.

2. CASE-BASED REASONING  
Case-based Reasoning (CBR) is an approach in Artificial Intelligence that uses previous experiences by adapting solutions in order to solve new problems [5]. CBR differs from other AI approaches since it does not rely on general knowledge of the domain and has the capability to acquire and store new experiences that can be referred to in the future. CBR mainly rely on specific knowledge of past experiences as new solutions to problems.

Acquired experiences in CBR are called cases. These cases are the ones recalled and applied to new situations. CBR works with the assumption that a similar game state or situation has been encountered in the past, and therefore that certain situation has a solution that might have been effective given that situation. Since the situations are not exactly the same, the CBR cycle allows the adaptation of the previous solution in order to fit the current situation [7]. The motivation of CBR comes from the way humans solve problems in that if a certain problem is encountered, some of the steps are recalled that have been done in a similar situation and then those steps that are potentially successful in the new situation are redone.
Generally, the CBR process can be described as a cyclic process comprising of the four Res (see Figure 1).

1. RETRIEVE the most similar case(s);
2. REUSE the case(s) to attempt to solve the problem;
3. REVISE the proposed solution, if necessary, to solve the problem; and
4. RETAIN the modified solution as a new case

Problem solving starts with an initial description of the problem. This defines the new case and this new case is used to retrieve a similar case from the collection of previous or stored cases also called as the case base. Through reuse, the retrieved case is combined with the new case into a solved case, or the proposed solution to the problem. This solution will be adapted to fit the current situation in the revision phase. A case also needs to be revised if it can not be reused or is not applicable to the current situation anymore. Once revised, the newly acquired case (revised case) will be saved and be part of the case base. This is the retention phase. For this paper the proponents will be focusing on a way of implementing the revision.

3. RELATED WORK

3.1 Case Based Tactician (CaT)

Case-based Tactician is the first case based reasoning system implemented in a real-time strategy game environment, designed to win against opponents. CaT is based on Wargus, a real time strategy game based on Warcraft II and built on top of Stratagus, an open-source real time strategy game engine.

CaT uses three knowledge sources: a state lattice, set of tactics for each state, and cases which map game states to tactics and their performance. The state lattice limits the number of tactics used for each state. A case is a 4-tuple ($C = \langle \text{BuildingState}, \text{Description}, \text{Tactic}, \text{Performance} \rangle$). BuildingState is the state of buildings, corresponding to a node in the lattice. Description is the set of features in the current situation. Tactic is the set of actions for a certain BuildingState, and Performance reflects the utility of choosing a Tactic for a BuildingState. CaT uses k-nearest neighbor and Euclidean distance for retrieval and does not need a fast indexing strategy because of the lattice. CaT relies on the game engine in order to adapt actions for reuse. CaT treats tactics as black boxes, and therefore repair is not present; only performance is updated.

3.2 Digma: A Role-Playing Game with Agent Plan Reformulation and Situational Reassessment

Digma is a role playing game that makes use of agents with plan reformulation and situational reassessment capabilities. The game is made up of team agents which are the human’s computer-controlled allies, and opponent agents, which try to stop the human player from reaching the goal to be achieved. In the game, there is a single learning agent, which is able to learn from experience. The learning agents decisions are based on situations previously encountered. All other agents are reactive agents, whose actions are preset for selected scenarios.

An experience represents the information learned during a certain situation. Good results coming from the agents’ decisions encourage agents to repeat the same actions in a future situation while bad results would allow the agent to learn and try a different approach when dealing with a similar situation in the future. Bad results lead to plan reformulation, when the goal is no longer attainable.

Agents also perform situational reassessment during the game. This done by identifying actions that are no longer applicable as well as performing other actions that could possibly lead to the goal instead. Reassessment allows the agents to grade the efficacy of their actions in situations encountered.

The learning agent makes use of decision trees in order to decide which action to execute given a certain state of the environment. Non-leaf nodes or decision nodes contain an index and features of the environment. The features discriminate the particular state of the environment. Leaf nodes contain an index as well as actions and plans. Situational reassessment is done on the leaf nodes only, when the current plan is no longer applicable. This is done by grading the previous plan first and adjusting it.

4. CAN SYSTEM

The research is implemented using an open-source real-time strategy game called Wargus that allowed the proponents to make use of a commercial and realistic game for the research.
The CAN system will be integrated into this game. It is an adversarial non-player character that is capable of learning strategies from the past actions of a human player and adapting the retrieved strategy online to fit the current situation. In order to limit the complexity of the system, it will focus on skirmishes or mini-battles in the game. The system will only come into play during those battles.

When a skirmish occurs, the system will first recognize the strategy of the human player by looking through the state of the environment and the units in the skirmish. The recognized strategy is defined as the combination of the type and number of the units, the formation of units used in the skirmish, as well as the terrain. Using the recognized strategy, the NPC will start the process of Case-based Reasoning and a possible counter-strategy will be selected against the human player. The counter-strategy will be similar to the recognized strategy that will include a sequence of actions that will tell the NPC where to move, where to hold position, which units or buildings to attack, and who will attack.

As mentioned before, CBR makes use of specific knowledge of past experiences. CAN will acquire this knowledge through training the NPC. Given manually-defined scripts that will play against a human player, the system will acquire the actions done by the human player. These will be the initial contents of the case base. Once enough strategies are acquired, the NPC can now play against the human player using the acquired strategies from previous experience. A mirroring of strategy will be done in such a way that the actions currently done by the human player are the actions done by the NPC during training and now the NPC will be performing the actions done previously by the human player. In doing so, the NPC is able to compete in a human-player level and will decrease the chances of predictability.

4.1 Case Representation

A case is represented as a series of states and actions. Each action has a pre-requisite state and an effect state (see Figure 2). The pre-requisite state is the state of the environment and the units inside the skirmish before doing the action. The effect state, structurally, is the same as the pre-requisite state. However, this state is captured after the action is done. In a sequence of actions, the effect state of the previous action is not necessarily the pre-requisite state of the next action. For example, one group of units is asked to attack the guard tower while another group of units is asked to attack a farm. These actions will be recorded sequentially as attacking the guard tower will be action 0 while attacking a farm will be action 1. Both actions already have pre-requisite state. While the effect state of action 0 is not yet captured since it has not yet damaged the guard tower. Therefore, the effect state of action 0 is not the pre-requisite state of action 1.

An action will always have a pre-requisite state but it is possible that an action will have no effect state. This happens when the user decides to do another action just right after commanding another action using the same units. For example, a group of units is asked to attack the guard tower, but before reaching the guard tower, the user changed the goal and decided to attack the town hall instead using the same group of units. Therefore, the action of attacking the guard tower does not have any effect state. Another instance for not having an effect state would be when the units to do actions died before even completing the action.

4.2 Adapting to the current situation

In a CBR process, once a similar case is selected, it will be executed. It is possible that during execution, the retrieved case is not applicable anymore to the current situation. Thus, revision is necessary. The same situation happens in the CAN system. While employing the retrieved strategy action per action, it is possible the NPC can no longer execute the action. This happens when the pre-requisite state of the action is not satisfied, meaning the current state is not similar to the pre-requisite state of the action. A state is considered similar if it passes the threshold after performing a similarity measure. Another instance that revision is needed is when, after doing the action and capturing the current state, it is not similar with the effect state specified in the case – a different result happened. Revision is also needed when the retrieved case has already run out of actions and there is still a skirmish.

Revision phase starts by looking through again in the cluster where the retrieved case resides. The cases are clustered by the strength of the friendly units relative to the strength of the enemy units. Therefore, the other cases belonging to the cluster where the retrieved case resides are considered similar cases. With these similar cases, a potential new sequence of action will be acquired and be executed. Basically, the system will suggest possible new actions for the NPC to do given its current situation.

In order to retrieve another sequence of actions, the system will be checking each pre-requisite state of each action in each case in the specified cluster. Each of these pre-requisite states
will be matched with the current state and if it passes the threshold then the system will be retrieving the set of actions starting from the similar pre-requisite state. If no pre-requisite state satisfies the threshold, then this means the system can no longer execute any actions and thus terminate the CBR process. The retrieved case will surely be considered as a failed case and its weights will be updated accordingly.

**Figure 3: Scenario**

For an example, given a scenario (see Figure 3), most of the units already have low HP after performing action 1. However in the Effect state of Action 1 in the case, the HP of all units is still high and the Enemy Buildings are decreased. Definitely, the actual effect state is not similar with the effect state in the case, this will call for revision. In the revision phase, the system will look into other pre-requisite state that is similar with the actual effect state. Here, the possible actions can be retreating or merging or groups (see Figure 4)

**Figure 4: Merging of Units**

Revision produces a new case. This new case will be composed of the actions coming from the retrieved case during the start of the CBR process and the actions from the cases retrieved from revision. Connecting old actions and new actions differs depending on the state that did not satisfy the current state (defective state): pre-requisite state or effect state. If the defective state is the pre-requisite state then all the actions starting from the first action reused to the action before the action with the defective pre-requisite state will be duplicated and will be included in the new case (see Figure 5). Unlike if the defective state is the pre-requisite state where the action with that state is no longer included, the case where the defective state is the effect state where the action is still included. However, it is replaced with the pre-requisite of the newly retrieved set of actions, the state that satisfied the current situation (see Figure 6).

Once the system has retrieved again a new series of action that has a similar situation as to when the previous case was preempted, the succeeding sequence of actions will be performed and if revision is needed again, then the whole process will again be repeated. This will continually be done until the end of skirmish.
5. IMPLEMENTATION ISSUES AND RESULTS

Before coming up with the design described above, the proponents encountered several issues. We initially thought of looking into the first pre-requisite of every case when looking for a new series of actions. However, we realized that most strategies reside on the middle of a skirmish; therefore, the most appropriate action would be in the middle of the case. For example, when the NPC needs to revise because its units are already decreasing, the most appropriate next action would be retreating. Retreating will not be found at the start of the skirmish but perhaps in the middle of almost at the last part of the skirmish. In order to increase the possibility of retrieving a series of action that describes retreating then we decided to look to every pre-requisite of the case.

Another issue encountered is how to connect the old case with the new set of actions and how to handle previous action executed. When implementing the actions, the system always monitors the pre-requisite and effect state. As seen in given in Figure 3, Pre-requisite State 3 cannot be checked until Effect State 2 is checked. This means that before doing the third action, the first action should already be finished. If for example Effect State 2 did not satisfy the current situation then even if Action 2 is already executed in the new case, this will not be included. Also we decided to disregard all states that have not been checked before revision. The new series of actions that will be connected will be reset, meaning the state ids will return to 0 and the action ids as well. A flag will indicate that that part of the case is the effect of revision. In doing so, when this new case is retrieved and while going through the action the system encounters the flag, this signals that everything the NPC did, like assigning the units to its target, will be change in accordance with the action. We decided to do this because of the complexity of making the old and new actions cohesive. Knowing that this new set of actions came from other cases, there will really be differences especially with the units that will implement the actions.

The proponents also decided not to choose the set of actions from cases that was retrieved before for the revision of the case as well as the case that was initially retrieved at start of the skirmish. This is to prevent repetition of actions that might cause cycling. Also in doing so, we are able to choose new actions that can provide a different strategy.

6. INITIAL TESTING AND RESULTS

In the initial testing done by the proponents with the system, the system was able to produce on the average of 10 cases per game coming. This came from the training done to the NPC. The number presented actually depends on the average number of skirmishes that occurs in a game. However, some skirmishes do not produce any case. The cases retained are only those cases wherein the human player actually the commanded and performed actions and not the actions performed by the embedded AI on any unit. Cases retained due of revision averages to also 10 cases per game. This is already expected by the proponents for initial testing mainly because there is still insufficient number of cases retained and each case still has a short series of actions.

CBR was said to be a slow algorithm because of the increasing number of cases. However, with the initial testing done, the NPC is still capable to reacting real-time with no lags during game play. The proponents also expect the system to continue performing well on this aspect despite the increase in cases main because of the clustering implemented. With clustering, search time is decreased.

7. CONCLUSION

To this date, the system is already working to the extent that is can already formulate new cases by adapting to the current situation. It has able to retrieved new actions to be performed and come up with a new strategy. However, the system is yet to be tested if it is able to retrieve the most appropriate actions to perform. Although the whole system has not yet been fully tested, considering the issues considered, it is expected that the system will be able to perform in such a way that it would bring more wins to the NPC as well as not making the same mistake that caused revision again.

8. REFERENCES


