AIIDE 2012 Doctoral Consortium
AAAI Technical Report WS-12-18

Applying Learning by Observation and Case-Based Reasoning to Improve Commercial RTS Game AI

Glen Robertson
Supervisor: Assoc. Prof. Ian Watson
Department of Computer Science
University of Auckland
Auckland, New Zealand
{firstname}@cs.auckland.ac.nz

Abstract

This document summarises my research in the area of Real-Time Strategy (RTS) video game Artificial Intelligence (AI). The main objective of this research is to increase the quality of AI used in commercial RTS games, which has seen little improvement over the past decade. This objective will be addressed by investigating the use of a learning by observation, case-based reasoning agent, which can be applied to new RTS games with minimal development effort. To be successful, this agent must compare favourably with standard commercial RTS AI techniques: it must be easier to apply, have reasonable resource requirements, and produce a better player. Currently, a prototype implementation has been produced for the game *StarCraft*, and it has demonstrated the need for processing large sets of input data into a more concise form for use at run-time.

Introduction

Despite increasing academic interest in video game Artificial Intelligence (AI) over the past decade, and a rapidly changing games industry which often competes on new technology (Laird and VanLent 2001), AI in video games has not improved as much as graphics, sound, animation or gameplay (Mehta et al. 2009; Tozour 2002). Adoption of academic research in AI is slow, likely because the industry considers it to be too impractical or risky to be applied in commercial games and because the underlying goals of academic and industry game AI often differ (Baekkelund 2006; Woodcock 2002).

Real-Time Strategy (RTS) is a genre of games which presents some of the toughest challenges for AI agents, making it an interesting area for academic research and a difficult area for developing competent AI. Even the best academic agents are still outmatched by experienced humans, while non-cheating industry agents are unable to provide a challenge to players past an intermediate level of skill, as they tend to exhibit predictable, inflexible behaviour (Baumgarten, Colton, and Morris 2009). In order to provide a challenge, agents are often allowed to "cheat" (given an advantage), but this can make the game less enjoyable if noticed by the players (Davis 1999).

Research Problem

The general problem examined by this research is the poor playing ability of commercial RTS game AI agents. This problem is broken down into two main sub-problems: the difficulty of creating RTS game AI (for both academia and industry), and the poor transfer of knowledge between academia and industry. The research aims to address these problems simultaneously by facilitating the creation of better RTS game AI suitable for commercial games.

Creating RTS game AI is a difficult problem, as it combines many different challenges which are difficult in their own right. These include reasoning at different levels of granularity, adversarial reasoning, long-term planning, a vast space of possible game states and player actions, long delays between actions and effects, randomness, hidden information, and real-time constraints (Buro and Furtak 2004; Laird and VanLent 2001; Mehta et al. 2009; Weber, Mateas, and Jhala 2010). The high degree of difficulty leads academic researchers to create complex AI agents which attempt to deal with the challenges but require large amounts of development effort. The simpler commercial game agents are unable to adapt to situations unforeseen by their developers, making them monotonous or easily exploitable by human players (Baumgarten, Colton, and Morris 2009; Tozour 2002).

AI used in commercial RTS games has remained largely the same over the past decade, generally using scripted behaviour with manually predetermined rules and behaviours. For example, the recent major RTS game *Starcraft II*[1], which was released over a decade after the original, still used a hand-made script-based AI (Sigaty 2008). One potential reason for this lack of change is the additional time and risk involved in attempting to use a new and more complex AI technique, particularly in a large and very time-pressured project like a commercial game (Baekkelund 2006). The risk is amplified if the AI is difficult to understand, test or debug, which is especially the case for non-deterministic techniques (Florez-Puga et al. 2009; Tozour 2002). Another reason may be the higher run-time resource usage of some techniques, which would reduce resources available to other elements of the game such as graphics (Baekkelund 2006). Also linked

[1] Blizzard Entertainment: StarCraft II: blizzard.com/games/sc2/

to this problem is the view that most game AI research is not practical enough to be used in commercial games, with most research only using games as a testbed (Champandard 2011). Finally, there is a difference in underlying goals in academia and industry: academia tends to aim to create strong players which are most capable of winning the game, while AI in commercial games aims to provide a fun challenge to the player (Baumgarten, Colton, and Morris 2009; Davis 1999; Tozour 2002).

Requirements

To carry out the aim of making it easier to create better commercial RTS game AI, an AI framework will be produced by this research. For the framework to address the problems set out above and be preferable to existing techniques used in commercial AI, the following requirements must be met:

- The AI produced by the framework must be at least as proficient at playing RTS games as existing AI in commercial games. It should also be more varied in behaviour, and less prone to exploitation by particular tactics which cause it to behave inappropriately. Ideally this would result in an AI which is more fun to play against, but measuring "fun" is beyond the scope of this work.
- AI should be approximately as easy to develop using the framework as it would be using the methods commonly used for commercial RTS games. This means it should be easy to test, debug and understand, and easy to apply to a range of RTS games, so that it could be quickly tried or used in a new game project. It should also be easily customisable, so it carries out particular behaviour in certain situations, to allow for story elements to be added.
- The AI should not be heavily resource intensive. Although it is unlikely to be as simple to run as existing methods, it should require a minority of the resources of a modern computer system.

Related Work

The work most closely related to this research topic lies in the areas of Learning From Demonstration (LFD) and Learning By Observation (LBO). LFD has been applied to RTS games already, but still requires significant development effort to apply, while LBO requires less development effort, but has not yet been applied to RTS games. To the author's knowledge, neither approach has been used in a commercial RTS game.

LFD seeks to shift the work required in creating AI from the developer to the agent, by allowing the agent to learn from examples of correct behaviour instead of being explicitly programmed with correct behaviour (Mehta et al. 2009; Ontañón et al. 2008). This should result in reduced development time required to create or modify an agent, and a wider range of agent behaviour, as the correct behaviour can be demonstrated more quickly than it can be programmed. It should also result in reduced debugging effort, as the correct behaviour can be demonstrated and incorporated if the agent is acting incorrectly in a particular situation (Mehta et al. 2009; Ontañón et al. 2008). However, the systems described by Mehta et al. (2009) and Ontañón et al. (2008) still require a significant amount of effort to annotate the demonstrated behaviour, in order to inform the LFD agent of the goals being undertaken by each action in the demonstration. This adds to the development effort required each time a new behaviour is demonstrated, but makes learning the correct behaviour easier for the agent and allows more direct control of the agent behaviour. It is also possible to automate much of the annotation process, but this still requires all of the possible goals an agent could be undertaking to be defined, which adds to the work required and results in a limited set of available goals (Weber and Ontañón 2010).

LBO is very similar to LFD, but emphasises that the agent purely observes the correct behaviour in each given situation, and is not given additional information to reveal any underlying reasoning (Floyd and Esfandiari 2011). This means that the agent must perform additional analysis in order to learn the appropriate behaviour in complex situations, but also means that no extra development effort is required to train an agent in new behaviours (Floyd and Esfandiari 2011). The Java Learning by ObservAtion Framework (jLOAF) allows LBO to be easily applied to a wide variety of situations because it uses a Case-Based Reasoning (CBR) framework which operates on generic inputs (observations about the current state) and outputs (actions in response to the current state) (Floyd and Esfandiari 2011). In order to apply jLOAF to a new domain, all that is required is an adaptor to convert from a state description to the generic input format, and from the generic output format to an action description (Floyd and Esfandiari 2011).

Recently Weber, Mateas, and Jhala (2012) and Jaidee, Muñoz-Avila, and Aha (2011) have also applied CBR to learn to play RTS games. They use Goal-Driven Autonomy (GDA) to create agents which form plans to achieve goals, and can dynamically respond to failures in the plan execution by choosing new goals and plans to compensate. Because significant domain knowledge is required for GDA, these systems both use learning to gather domain knowledge – Weber, Mateas, and Jhala (2012) uses LFD from game logs (replays) while Jaidee, Muñoz-Avila, and Aha (2011) learns by playing the game.

Research Plan

In order to address the research problem and meet the requirements set out above, LBO and CBR will be used in a generic framework which can be easily adapted to different games. CBR is effective at working with missing information and uncertainty (Floyd and Esfandiari 2009), and given sufficient training data, it should be able to behave correctly in a wide range of situations. LBO will allow an agent to be trained by simply demonstrating correct play, thus making such an agent easy to create. To apply the framework to a new game, all that is required is to adapt the user input and game state output into a generic format which is understood by the CBR system. This will build upon the work on jLOAF (Floyd and Esfandiari 2011), customising the framework to specialise in RTS gameplay.

To test this approach, it will be applied to *StarCraft*[2], using replays of expert players to train the agent. After this initial implementation, there are a number of areas which may be investigated for improvement. Firstly, and most importantly, a method for automated feature weighting and case generalisation will be necessary in order to reduce the vast amount of information available and extract useful knowledge. Next, investigation into optimising the case base structure for traces should decrease retrieval time and resource usage. Tools will also be added to aid in understanding and debugging of the agent's actions. Finally, the agent may be integrated with reinforcement learning in order to produce better short-term tactical behaviour, or with case-based planning for better long-term strategic behaviour.

At each stage, the agent will be tested against the built-in *StarCraft* AI tools could be provided to aid in understanding and debugging the agent's actions and evaluated against the requirements.

Progress

At this stage the jLOAF framework has been applied to *StarCraft* using the processed dataset from Gabriel Synnaeve's[3] "bwrepdump" tool as training data. This has produced an extremely large number of cases (approximately 90 million) which must be reduced to a manageable size in order to be used in real-time. This highlights the need for generalisation of the case base in order to combine the information of many similar cases into a few representative cases, and optimisation of the case base in order to make the relevant cases quickly accessible during game play. In order to produce a working prototype, a subset of the data is being used and many features are being excluded. The next stage will be automating the optimisation process to allow for large numbers of cases to be processed into a more concise set of representative cases without additional human input to decide which cases are similar or unique.

References

Baekkelund, C. 2006. Academic AI research and relations with the games industry. In Rabin, S., ed., *AI Game Programming Wisdom*, volume 3. Boston, MA: Charles River Media. 77–88.

Baumgarten, R.; Colton, S.; and Morris, M. 2009. Combining AI methods for learning bots in a real-time strategy game. *International Journal of Computer Games Technology* 2009:10.

Buro, M., and Furtak, T. M. 2004. RTS games and real-time AI research. In *Proceedings of the Behavior Representation in Modeling and Simulation Conference*, 63–70. Citeseer.

Champandard, A. J. 2011. This year in game AI: Analysis, trends from 2010 and predictions for 2011. http://aigamedev.com/open/editorial/2010-retrospective/. Retrieved 26 September 2011.

Davis, I. L. 1999. Strategies for strategy game AI. In *Proceedings of the AAAI Spring Symposium on AI and Computer Games*, 24–27.

Florez-Puga, G.; Gomez-Martin, M.; Gomez-Martin, P.; Diaz-Agudo, B.; and Gonzalez-Calero, P. 2009. Query-enabled behavior trees. *IEEE Transactions on Computational Intelligence and AI in Games* 1(4):298–308.

Floyd, M., and Esfandiari, B. 2009. Comparison of classifiers for use in a learning by demonstration system for a situated agent. In *Workshop on CBR for Computer Games at the International Conference on Case-Based Reasoning (ICCBR)*.

Floyd, M. W., and Esfandiari, B. 2011. A case-based reasoning framework for developing agents using learning by observation. In *Proceedings of the IEEE International Conference on Tools with AI*, 531–538.

Jaidee, U.; Muñoz-Avila, H.; and Aha, D. 2011. Case-based learning in goal-driven autonomy agents for real-time strategy combat tasks. In *Proceedings of the Workshop on CBR for Computer Games at ICCBR*, 43–52.

Laird, J., and VanLent, M. 2001. Human-level AI's killer application: Interactive computer games. *AI Magazine* 22(2):15–26.

Mehta, M.; Ontañón, S.; Amundsen, T.; and Ram, A. 2009. Authoring behaviors for games using learning from demonstration. In *Workshop on CBR for Computer Games at ICCBR*.

Ontañón, S.; Mishra, K.; Sugandh, N.; and Ram, A. 2008. Learning from demonstration and case-based planning for real-time strategy games. In Prasad, B., ed., *Soft Computing Applications in Industry*, volume 226. Springer Berlin / Heidelberg. 293–310.

Sigaty, C. 2008. Blizzard answers your questions, from Blizzcon. http://interviews.slashdot.org/story/08/10/15/1639237/blizzard-answers-your-questions-from-blizzcon. Retrieved 13 June 2012.

Tozour, P. 2002. The evolution of game AI. In Rabin, S., ed., *AI Game Programming Wisdom*, volume 1. Hingham, MA: Charles River Media. 3–15.

Weber, B., and Ontañón, S. 2010. Using automated replay annotation for case-based planning in games. In *Workshop on CBR for Computer Games at ICCBR*.

Weber, B.; Mateas, M.; and Jhala, A. 2010. Applying goal-driven autonomy to starcraft. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment (AIIDE) Conference*, 101–106. AAAI Press.

Weber, B.; Mateas, M.; and Jhala, A. 2012. Learning from demonstration for goal-driven autonomy. In *Proceedings of the AAAI Conference on AI*, 1176–1182.

Woodcock, S. 2002. Foreword. In Buckland, M., ed., *AI Techniques for Game Programming*. Premier Press.

[2]Blizzard Entertainment: StarCraft: blizzard.com/games/sc/

[3]Gabriel Synnaeve's homepage: emotion.inrialpes.fr/people/synnaeve/