Neural Networks for Small-Scale Combat Optimization in Real-Time Strategy Games

Jacob Menashe and Vineet Keshari

December 9, 2012

Abstract

A major challenge for players and AI modules alike in the real-time strategy community is the effective control of individual units in combat scenarios. Starcraft, a real-time strategy game released by Blizzard Entertainment, provides built-in heuristics for assisting players in combat management, but these heuristics are limited in their ability to make full use of a unit's potential. We present a series of techniques for training neural networks in small-scale combat situations, and lay the groundwork for further improvements to this system. We present a neural network design for controlling combat units, which, when coupled with the NEAT evolutionary learning algorithm, is able to consistently overcome Starcraft's built-in AI. In most cases our technique achieves optimal champion performance in less than 100 generations, and scales to large battles in a variety of unit configurations. Our system achieves a X% win rate against the AI, and a X% win rate against humans in small-scale battles.

1 Introduction

Micro-management is a major component of real-time strategy (RTS) games such as Starcraft. Players must effectively move individual units around for gathering resources, constructing buildings, or for tactical combat. In the case of tactical combat, micro-management requires quick action on the part of the player and thus can be difficult to master even with a small number of units. While modern games tend to minimize the degree to which players must control units on a small scale through the implementation of batch commands, this is usually achieved with simple heuristics a significant amount of potential is sacrificed for such enhancements. For a simple example of this, consider the following scenario: a player might take a heterogeneous group of close-range and long-range units and issue a batch attack command, at which point her units will either attack at random or attack one unit at a time. A more effective strategy might be to allow ranged units to disengage and re-engage based on the enemy's response, to allow these units to keep their distance, while close-range units focus on the greatest threat to allied ranged units. In this situation, the simplicity of the batch AI has diminished the effectiveness of the player's army. Such circumstances are prevalent in modern RTS games.

The domain of micro-management in combat environments is therefore well-suited to the application of intelligent, autonomous agents. While the best AI implementations are still unable to compete with most humans X in terms of high-level planning and strategy, a trained unit controller agent would have little difficulty outperforming a human competitor for any reasonably-sized army. Indeed, while an agent may potentially make hundreds of decisions and unit actions per second, even the most skilled human Starcraft players can only achieve about 5 X.

In this paper we present a novel application of NEAT to small-scale combat problem, restricted to the domain of Starcraft. We devise a variety of network topologies and configurations for outperforming the game's built-in AI, as well as human players, and describe how our technique may be extended for outperforming winning bots from the Starcraft AI Competition X. In Section 2 we discuss some background to AI development in real-time strategy games and specifically in Starcraft, and examine another body of work that focuses on small-scale combat. In Section 3 we review our approach to learning controller agents, along with a discussion of software libraries and implementation challenges in Section 4. We evaluate our results agains the built-in Starcraft AI, against other bots from the Starcraft AI Competition, and other humans in Section 5. In Sections 6 and 7 we conclude and discuss additional updates and improvements to our method, toward the ultimate goal of providing a general-use combat manager.

2 Background

3 Technical Approach

4 Software

5 Experiments

6 Conclusions

7 Future Work

References