Zerteross Neural Network Based StarCraft AI Project Code Documentation

1 System Requirements

It is recommended to use Windows XP SP3. Using Windows 7 may result in some unknown issues. For a Mac user, please install a virtual environment because Mac version of StarCraft does not really work on the latest Intel architecture and also BWAPI requires a Windows environment for developing.
Also Visual C++ is required for compile the BWAPI Version 3.7.4.

2 Components

Our code contains two major components:
BWAPI neural networks based AI module (repository name: zerteross):
The code for AI module is responsible for controlling ally units in the StarCraft game using neural networks. In order to generate and run the neural networks evolved by NEAT, the AI module contains come code from NEAT C++.

NEAT C++ (repository name: neat):
This part of code contains a revised version of NEAT C++. The NEAT C++ is originally developed by Kenneth Stanley in UT Austin(http://nn.cs.utexas.edu/?neat-c). The source code of NEAT C++ is written on Unix/Linux platform. We fix some minor bugs when port to VC++ project.

3 Environment setup

3.1 Installing StarCraft
(1) Register a Battle.Net account on http://us.battle.net/en/
(2) Buy and download StarCraft:BroodWar from http://us.blizzard.com/en-us/games/sc/
(3) Install StarCraft
(4) Update StarCraft to 1.16.1 (You may do this through connecting to Battle.Net in the game)

3.2 Installing BWAPI
(1) Download BWAPI from https://code.google.com/p/bwapi/. For compatibility with most AI from other research group, we use version 3.7.4 instead of the newest beta version 4.0.1B.
(2) Install BWAPI following the instruction: https://code.google.com/p/bwapi/wiki/UsingBWAPI

3.3 Porting NEAT C++ to Windows platform
(1) We use a cross platform make tool called CMake to generate Visual C++ NEAT project from the GCC version. Download from http://www.cmake.org
(2) In the CMake GUI, specified the right VC++ version, configure and generate VC++ project.
(3) Build the NEAT VC++ project, and will get a neat.exe executable.

4 Running AI Module

4.1 Configuration

(1) Configure the settings in zerteross\ConfigurationFiles\nncontroller_config.ini. For running the AI Module we only care about the settings under [General] and [nncontroller] sections.
(2) Open the AI Module project in Visual C++ 2008 Express.
(3) In ExampleAIModule.cpp line 48, specified the path of nncontroller_config.ini.

4.2 Compiling and running

(1) Build the project in Debug mode and copy the generated ExampleAIModule.dll to StarCraft \bwapi-data\AI\. Rename it as ExampleAIModule.dll.
(2) Place appropriate population file (zerteross\ConfigurationFiles\AIModule\winnierOrganisms\) to the path specified in nncontroller_config.ini.
(3) Run StarCraft\BWAPI 3.7.4\Chaoslauncher\Chaoslauncher.exe, and check BWAPI Injector (1.16.1) DEBUG.
(4) Config Chaoslauncher by click the Config button
(5) Start the game by click the Start button

5 Training NN based agent

5.1 Configuration

(1) Open the NEAT project in Visual C++ 2008 Express.
(2) In experiments.cpp line 23, specified the path of nncontroller_config.ini.
(3) Place appropriate file, which contains start genes for generating the initial population, in the path specified in nncontroller.ini. (The start genes file can be found in corresponding folder under zerteross\ConfigurationFiles\NEAT\)
(4) Configure the settings in zerteross\ConfigurationFiles\nncontroller_config.ini.

5.2 Compiling and running

(1) Build the NEAT project and get the neat.exe executable.
(2) Place appropriate neat config file (*.ne) in the same folder with neat.exe. Neat config file can be found in corresponding folder under zerteross\ConfigurationFiles\NEAT\.
(3) Run the NEAT in command line using command: neat config.ne. When some choices come up, choose 0.
(4) Run the StarCraft AI Module using Chaoslauncher (the same as 4.2)
(5) The training results will be exported to the files specified in nncontroller_config.ini.

6 Run multiple instance of StarCraft AI Module

The instructions can be found in https://code.google.com/p/bwapi/wiki/MultiInstance.