COMP30290 Natural Computing Project Proposal

Author Name: Kimberley Manning

Author ID: 12251405

Project Title: Domain independence in natural algorithms

What is the question that this proposal addresses?

Games have long been one of the testing grounds for new artificial intelligence research, and the application of natural computing methods is no exception. Games, particularly well-studied ones such as Ms. PacMan, present a well-understood problem and an opportunity to compare new methods against the current best approaches on an even playing field. The best strategies are well-known and it is common to see strategies which are learnt or evolved by each new algorithm compared to these.

Programmers commonly provide their algorithms with information that will assist in developing these strategies. For example, in Ms. PacMan a common strategy is for Ms. PacMan to wait for all the ghosts to approach her, then consume a power pill and chase the ghosts while they are nearby. Therefore, Ms. Pacman needs to know the distance to each ghost and the distance to the nearest power pill. The programmer therefore hand-codes functions to calculate these. Programmer time and effort, and familiarity with the problem domain, is required, and this is before going into the issues with grammatical evolution and genetic programming algorithms described in the next section.

On the other hand, much research has been done with neurologically-inspired algorithms in game AI, where the focus is on the neural network 'learning' the rules of the game with a minimum of prior knowledge. This project will attempt to determine whether grammatical evolution can be used to generate a Ms. PacMan controller given only generic starting information, and how such an approach compares to one using neural networks. The minimum the controller needs is a sensor (to take in information about the world) and actuators (to act on the decisions it makes). For a 2D game such as Ms. PacMan, this means functions to read a game board array, and to send the commands Left, Right, Up and Down back to the interface.

Why is this problem significant?

There are a number of challenges in working with evolved programs, such as those produced by grammatical evolution.

In many cases solutions generated by natural computing algorithms are not particularly successful against hand-crafted solutions. Much programming time is devoted to, firstly, working out which of these subproblems are most important to the task, and secondly, hand-coding solutions. This is even before the often significant amounts of time needed for the learning or evolutionary stage of development.

At the end, the programmer is left with code only suited for one domain. Developing code for a problem that is not as well-understood as Ms. PacMan still requires a not-insignificant investment by the programmer in order to understand the problem, develop an idea of possible ways of solving the problem, and artificially break this down into helper functions that the natural algorithm can use to generate a solution.

Grammatical evolution does not produce programs that meet any of the traditional 'code quality' criteria: for example, readability, maintainability, extensibility, or security. It is obvious that GE is not designed to produce solutions which are to be kept long-term: they are throwaway solutions, designed to be used for the immediate problem but not examined, modified or extended later.

A related reason for applying natural computing methods to well-understood problems such as Ms PacMan is to see if novel solutions are generated. Once the principle of the new solution is understood, it can then be applied in hand-crafted solutions. The more domain knowledge is known from the beginning to the algorithm, however, the closer the final solution is likely to be to a hand-crafted one.

If GE can be successfully used with a minimum amount of domain-specific knowledge, many of these issues become irrelevant: less time and mental effort is required for initial setup, novel solutions to old problems have more freedom to evolve, and code requires minimal effort to transfer to a new domain (all that is required is to write an interface to the new problem world). GE could be applied in fields where human programmers have limited knowledge of best strategies.

How will the question be addressed?

Experimental Method

Domain independence means the 'sensors' and 'actuators' available to Ms. PacMan should be applicable to other problems in its category (2D grid-based games/simulations using the arrow keys as input). The game software used will be the Java code used in the 'Ms Pac-Man vs Ghosts League' at pacman-vs-ghosts.net. This will be passed through an interface which converts the game information to an array representation of the current map, with each array location containing information about what is currently at that point (for example a ghost, wall or power pellet). The simulation packages GEVA and SNNS will be used to develop the grammatical evolution and neural network controllers respectively.

Timeline

Week 6: Literature search, particularly on neural networks

Week 7-8: Coding of interfaces, setup of simulations and other support functions

Week 9-10: Simulations

Week 11: Compilation of report

Week 12: Final work on report (project due Thursday)