Project Proposal:
"Evolution and Learning of Cooperation in a Competitive Environment"

John Richter (602R3846)
Supervisor: Philip Sterne

13/03/2005

1 Statement of the Problem

There exist many examples of the use of artificial intelligence to find the most appropriate strategy to use in a competitive environment. This is due to artificial intelligence's high quality solutions to possible NP type problems. Unfortunately, few previous examples accurately model true business and social environments, where some form of cooperation is required for the group to survive despite every individual (or sub-group) striving in competition to be the ultimate winner. The problem that I choose to address is that of modelling competition in an environment where cooperation is required between entities in order for all to survive, in such a way that it is easily extended to suit particular situations in the real world.

Further difficulties arise when it is considered that strategies are constantly evolving and changing, dependant upon circumstances and flexibility of the decision-maker. This makes genetic algorithms inappropriate, as they only change strategies at 'birth', whereas these models would require strategies that can mutate and change during the lives of the agents.

2 Feasibility Study

This project will require research into the areas of genetic (or evolutionary) algorithms, as well as insight into their use in modelling interaction between

agents. Rhodes University library can provide all the required literature, while advice and insight have been offered by my supervisor, Philip Sterne.

2.1 Literature

An initial survey of the literature yielded the following resources:

- *"The Evolution of Cooperation"* by Robert Axelrod, New York, Basic Books, Inc, 1984
- The Masters' thesis of Clinton Brett McLean, *"Design, Evaluation and Comparison of Evolution and Reinforcement Learning Models"*, Rhodes University, 2001.

I expect to make frequent use of the following websites:

- "*AIGuru"* (http://www.aiguru.com/, 2003)
- *"GA Archives"* (http://www.aic.nrl.navy.mil/galist/, 2004)
- *"Game Theory.net"* (http://www.gametheory.net/, 2004)

2.2 Electronic Resources

In terms of hardware, this project will require a reasonably powerful computer to run through the many iterations of strategy that will be required to be tested. This has already been provided by Rhodes University.

Many modern programming languages are appropriate for the implementation of this project, but this researcher feels most comfortable using C-like languages, programming on a GNU/Linux platform.

3 Outcomes and Deliverables

The main deliverable will be a program modelling cooperative/competitive behaviour.

In order to display the use of the algorithms studied to model competitive/cooperative behaviour, this researcher has selected to model a situation based on the popular television series *"Survivor"*, produced by CBS (http://www.cbs.com/primetime/survivor/
The deliverable project will be a program modelling this situation, including a

report on the most successful strategies used, anomalies discovered, and documentation on further insights discovered throughout the course of the research.

This program will model the game in the following ways: Several agents will be programmed to begin with random strategies, and shall need to provide essential resources for the group ("food"), while also focusing on political strategies. Through a sequence of democratic voting (modelling economic or political strategies), the least desirable agents shall be removed from the game.

A successful strategy can be determined by the number of successful games a particular agent has won, and, to a lesser degree, how many time periods the agent has remained in a game.

4 Possible Extensions

The main extensions to this project are future models of real-world situations. The ability to accurately model real-world strategies, alliances, dependencies, and other political constructs, and then derive outcomes or consider strategies is very applicable to business, economics, politics, and a host of other fields.

A possible extension into a real-world situation would be to model several businesses competing in a market that requires some or all of them to co-exist, due to complementary or dependant products.

5 Significant Algorithms/Structures

Two significant implementations of Genetic principles must be taken into account when considering this study: the structure of the agents themselves (as they will need to be complex enough to be able to model real-world entities, but not so complex that they include superfluous data) and the algorithms used for agent learning (as it must allow a combination of long- and short-term feedback, and be effective at rapidly approaching satisfying strategies).

Firstly, the agents themselves shall be objects, with several important properties: an array of political opinions of other agents, a level of 'aliveness' (modelling success, or hunger), and a construct representing strategy. Strategy shall need to take into account threat levels in other agents (allowing paranoia to be modelled), trust (in order to model alliances), 'neediness' (allowing good providers to not be killed off early by their more political counterparts), and finally some small amount of randomness to break ties. The actual construct will

not only hold this information (which will essentially be the same for any given agent in the same position), but most importantly 'weightings', the subjective, interpreted opinion of each agent on how to react to the situational variables.

Secondly, the learning mechanisms used by the agents are of particular interest. They shall be using a combination of genetic algorithms and reinforcement learning, in order to increase the speed at which learning takes place and to better model real-world strategy and decision-making. The best agents shall have their weightings combined through variable interpolation (crossover, "sex"), in order to find the best combination of starting weights. Some randomness will be included to stop local maximums becoming a problem. Every time period ("vote"), the agents shall observe the outcomes of the opinions of the other agents (the outcome of the votes), and alter their weightings accordingly.

6 Proposed Timeline

Please note that this proposed timeline is conservative in time required for the tasks.

Time Required	Activity
4 weeks	Reading relevant literature
2 weeks	Implementing agents structure
3 weeks	Building model and environment
1 week	Testing agents in environment
2 weeks	Modifying and testing learning algorithms
4 weeks	Testing strategies and learning models