Cross-Validation of Fitness Scores During Co-evolution Using the ’Trap-the-Cap’ Board Game as a Testbed

C J Flynn

26 October, 2009

Department of Computing
Faculty of Mathematics, Computing and Technology
The Open University

Walton Hall, Milton Keynes, MK7 6AA
United Kingdom

http://computing.open.ac.uk
Cross-Validation of Fitness Scores During Co-evolution
Using the 'Trap-the-Cap' Board Game as a Testbed.

A dissertation submitted in partial fulfilment
of the requirements for the Open University's
Master of Science Degree
in Computing for Commerce and Industry.

Colin James Flynn
(X2106621)

9 March 2010

Word Count:
14,691
Preface

I would like to thank my supervisor George Haywood for the time he spent providing guidance and comments on my work. I would also like to thank my family, Jan, Lorna and Emily for their patience.
# Table of Contents

Preface........................................................................................................................................i
Table of Contents.........................................................................................................................ii
List of Figures...............................................................................................................................iv
List of Tables................................................................................................................................v
Abstract.........................................................................................................................................vi

1 Introduction...............................................................................................................................1
   1.1 Trap-the-Cap.......................................................................................................................1
   1.2 Problem Overview..............................................................................................................2
   1.3 Aim of Proposed Research...............................................................................................7
   1.4 Research Question............................................................................................................8
   1.5 Contribution to Knowledge.............................................................................................8

2 Literature Survey.....................................................................................................................10
   2.1 The Reinforcement Learning Method..............................................................................10
   2.2 Evolution of Fixed Neural Network Architectures......................................................11
   2.3 Evolution of Neural Network Architectures..................................................................11
   2.4 Neuro-Evolution of Augmenting Topologies (NEAT).....................................................12
   2.5 Evolutionary Arms Races and Co-evolution.................................................................13
   2.6 Fitness Sharing................................................................................................................14
   2.7 The Hall of Fame and Co-evolutionary Memories..........................................................16
   2.8 Cross-Validation and the Measurement of Progress.......................................................17

3 Research Methods..................................................................................................................19
   3.1 Project Framework...........................................................................................................19
   3.2 Evaluative Measurements..............................................................................................21
   3.3 Cross Validation Assessment..........................................................................................23
   3.4 Implementation.................................................................................................................24

4 Experiments.............................................................................................................................30
   4.1 Introduction......................................................................................................................30
   4.2 Parameters used for Evolution.......................................................................................30
   4.3 Trap-the-Cap Experiments.............................................................................................32

5 Analysis of Results..................................................................................................................35
   5.1 Co-evolution.....................................................................................................................35
   5.2 Trap-the-Cap games........................................................................................................41
5.3 Network Complexity ................................................................. 42
5.4 Dominant Strategy Play-off’s ..................................................... 43
5.5 Best of Run Play-offs ............................................................... 47
6 Discussion and Conclusions ...................................................... 50
References ............................................................................... 53
Index ................................................................................... 57
Appendix A: Extended Abstract ............................................... 59
List of Figures

Figure 1: Trap-the-Cap board layout........................................................................................................2
Figure 2: High level class diagram of Co-evolution software.................................................................25
Figure 3: Outline of core co-evolution algorithm.....................................................................................26
Figure 4: Mating process for two sets of network links.................................................................28
Figure 5: Typical Set of Experimental Parameters..............................................................................31
Figure 6: Best of Generation scores for populations A and B...............................................................35
Figure 7: Best of Generation scores for populations A and B (Expanded scale)...............................36
Figure 8: A poor start for populations B..............................................................................................37
Figure 9: Dominant Strategies for all 8 experiments............................................................................37
Figure 10: Compatibility distance from seed, experiment 1..............................................................39
Figure 11: Detail from 'Compatibility distance from seed, experiment 1'..........................................40
Figure 12: Number of links in dominant strategy networks..............................................................42
List of Tables

Table 1: Types of co-evolutionary progress ......................................................... 17
Table 2: Comparison Method Summary ............................................................... 23
Table 3: Experiments with up to 240 moves per player ................................. 33
Table 4: Experiments with up to 120 moves per player ................................. 33
Table 5: Dominant strategy play-off scores (Expt 3 vs Expt 2) ......................... 44
Table 6: Dominant strategy play-off scores (Expt 4 vs Expt 2) ......................... 45
Table 7: Dominant strategy play-off scores (Expt 5 vs Expt 2) ......................... 46
Table 8: Dominant strategy play-off scores (Expt 8 vs Expt 6) ......................... 47
Table 9: Master Tournament Winners ............................................................... 48
Table 10: Best-of-run play-off, ( 240 moves per player) ................................. 48
Table 11: Best-of-run play-off, ( 120 moves per player) .................................. 49
Abstract

Games have always been used as a convenient way of testing AI techniques, they have well defined rules and well defined outcomes.

The reinforcement learning method of co-evolution is investigated using the board game of Trap-the-Cap. Co-evolution is used when no teacher is available for game playing 'agents' to learn from. Essentially, two populations of agents take it in turns to rank each other before mutating and hence evolving. The populations start out as completely naïve Trap-the-Cap players and gradually increase in sophistication over the ensuing generations.

A criticism of co-evolution is that each population of agents is used for training and testing the other population. This is normally to be avoided, but this is not easy for co-evolution which was selected because no teacher was available to provide training.

This thesis investigates the technique of injecting independent test agents into the co-evolution cycle to provide 'Cross-Validation' of the ranking of a population. It asks the question 'does the use of Cross-Validation provide measurable benefits in terms of speed of evolution, network complexity and performance?'

Neural networks were used as the agents in the two populations. The particular technique used to evolve and mutate them is called Neuro-Evolution of Augmenting Topologies (NEAT) which is a method that allows neural networks to use the cross-over operation as well as mutation. Cross-Validation was achieved by providing a source of independently evolved neural networks as an additional source of testers during the co-evolution cycle.

Results were encouraging and showed that there was indeed an advantage gained in terms of performance, speed of evolution and network complexity. However, these effects were only present for the first one hundred or so generations, after which the advantage disappeared. This may have been related to the game of Trap-the-Cap itself and the parameters used to evolve players.
1 Introduction

This research project will use a board game called Trap-the-Cap as a vehicle for evaluating techniques from the field of artificial intelligence (AI). Games have been used in the study of artificial intelligence from its earliest days (Samuel 1959). Samuel demonstrated machine learning techniques using the game of Chequers and in his paper gave the following reason;

“A game provides a convenient vehicle for such study as contrasted with a problem taken from life, since many of the complications of detail are removed.”

Another good reason, is the ability of different approaches to be directly compared since a game has known rules that can be easily replicated across different research groups. In essence, a game can be thought of as a protocol for the comparison of different algorithms. This section will give an overview of artificial intelligence, genetic algorithms and neural networks relevant to the proposed project title.

1.1 Trap-the-Cap

Firstly though, a description of the rules for the board game Trap-the-Cap; see figure 1 for a picture of the board. There are 2-6 players who each have 6 caps placed on their home base and take it in turns to move a cap. A cap is moved by throwing the die, selecting a cap and moving it in any direction by the number of spaces indicated on the die. The direction of movement cannot be altered during a move and only one of a players caps (or stack of caps) may be on any given space, apart from his home base. If a players cap moves into a space already occupied by an opponents cap, or stack of caps, then he can place his cap over it and capture it. To keep trapped caps and remove them from play, they must be moved back to a players home base using his normal turns. Once a stack of caps has returned to a players home base, any caps belonging to that player are available for play in the normal way; any caps not belonging to that player are permanently removed from play. Some spaces on the board are coloured grey, these are safe spaces which can be occupied by up to three caps (or stacks) of different colours without risk of being 'capped'. Players can agree different endings, for example the player with the most trapped caps wins, or the last player with caps remaining on the board wins.
1.2 Problem Overview

The field of computer based Artificial Intelligence (AI) covers a wide range of methods and techniques, all with the aim of producing systems, that to some extent or other, are able to perceive, reason and act. The concept of a 'rational agent' can be used as the basis upon which to build and classify the many different branches of the AI discipline (Russell & Norvig 2003). Russell and Norvig define a rational agent as

“One that acts so as to achieve the best outcome, or when there is uncertainty, the best expected outcome.”

This very general view encapsulates all AI solutions as having, in some form, the ability to perceive an environment using 'sensors' and some means of influencing the environment using 'actuators'. The various sub-fields of AI provide the bit in between the sensors and the actuators; the reasoning, and possibly some learning as well.

An AI agent need not have the ability to learn anything about its environment. Learning does not matter if the environment is known and unchanging; an agent can be equipped with enough initial knowledge to perform its tasks satisfactorily. However, if an environment is not stable, then an AI agent has to be capable of learning about and responding to changes. The type of learning an agent is capable of is sub-divided into supervised, unsupervised and reinforcement learning methods (Russell & Norvig 2003). These three sub-divisions are defined by the type of feedback available for learning. Supervised learning requires a 'teacher' to provide feedback on an agent's actions so that it can learn to correct itself; however, it is not always the case that a teacher exists with the

**Figure 1: Trap-the-Cap board layout**
necessary knowledge. Unsupervised learning agents can learn to recognize patterns in
input information but cannot learn what to do, as they have no concept of what constitutes
a correct action. An agent that uses reinforcement learning is one where learning is driven,
not by a teacher, but by rewards gained from the agents performance. Reinforcement
learning is important in game playing problems since reward can be easily defined, for
instance as win, draw or lose, whilst absolute knowledge of a correct move is not
necessarily available. Trap-the-cap could be played by a supervised learning agent or a
reinforcement learning agent; this research project will pursue the reinforcement learning
route since it requires no knowledge of optimum play, knowledge that does not readily
exist.

Using the term, 'reinforcement learning' requires some care; there is the Reinforcement
Learning (RL) method, which makes use of a Temporal Difference algorithm that assigns
credit for the many steps along the way to success (or failure), as exemplified in the book
by Sutton and Barto (Sutton & Barto 1998), and then, there are methods that use a
feedback signal to reinforce desired behaviour. One such group of methods are genetic
algorithms. In their paper “Genetic reinforcement learning for neurocontrol problems”
(Whitley et al. 1993), the authors state that;

“genetic algorithms are, in a sense, inherently a reinforcement learning
technique; the only feedback used by the algorithm is information about the
relative performance of ...different potential solutions.”

What would be a good way of proceeding in Trap-the-Cap given that there is no accepted
way of optimal play? One avenue that has demonstrated some success, is to use genetic
algorithms to evolve a neural network. In their paper 'Evolution, Neural Networks, Games
and Intelligence' (Chellapilla & Fogel 1999), the authors state that;

“The evidence presented indicates that there is a profitable synergy in utilizing
neural networks to represent complex behaviours and evolution to optimize
those behaviours, particularly in the case where no extrinsic evaluation
function is available to assess the quality of performance.”

This thesis will therefore pursue a study of neural networks that learn through an
evolutionary process governed by a genetic algorithm.

A genetic algorithm (Mitchell 1998) borrows from the concept of natural evolution; a
chromosome, is a collection of genes, and the genes (genotype) provide the instructions to
generate a physical incarnation of the organism (phenotype). If the organism is successful
in its environment, then it may breed with other members of its species to produce offspring which, through inheritance of 'good' genes, may be better adapted. In computers, the genes are usually represented by bit strings or real numbers or any representation that fits the problem. A population of chromosomes is generated and each one assessed for 'fitness'; this assessment can be carried out either by an evaluation function, which can determine directly from the chromosome how good a solution it will produce, or by physically generating agents from the chromosome population and observing which are the fittest. The chromosomes of the fittest agents reproduce amongst themselves using crossover and mutation. Crossover can splice parts of different parent chromosomes together to produce new chromosomes, which can also in turn have random genes changed or mutated. Over successive generations the agents generated from the chromosomes become better fitted to their environment. It is very important to maintain a diverse population of agents to allow the strongest to develop general game playing strategies.

A neural network (Gurney 1997) (loosely inspired by the neurons in an animal brain) is an interconnected assembly of simple processing elements called nodes; each node sums any inputs received and when a threshold is crossed, produces an output. A neural network will be provided with input signals from the environment, for example, values of image pixels in a pattern classification problem, which will be directly connected to some of the nodes in the network. Output connections from one or more nodes determine which action to take, for example which category an image should be placed in. Each connection between a node pair connects an output to an input and has an interconnection strength associated with it; this can be thought of as amplifying the signal from the output node before it reaches the input of the second node. The processing ability of the network is stored in the inter-node connection strengths, or weights. A node can have any number of inputs from other nodes and have its output connected to any number of other nodes, in any configuration. Learning is achieved by altering the interconnection weights so that input signals applied to the input connections are transformed to the correct pattern of output signals on the output connections. Traditionally, learning has taken the form of having training data that represents perfect output for a given input. The difference between training data and the actual output is used to adjust the network weights.

So, a neural network that learns through an evolutionary process can be implemented as
follows. The genotype would be a collection of genes representing the network components, connections, connection strengths etc. The phenotype (neural network), or agent would then be generated from the genotype and be used to play games against other players. Each agent is said to represent a 'strategy’, a way of playing the game represented by the network and its connection strengths. The fittest and best neural network game playing agents would then be selected and reproduction of their genotypes undertaken.

The question arises of how to represent the Trap-the-Cap board as inputs to a neural network and what sort of output should be generated? Network inputs should, as a minimum, represent the position of every 'cap' and its status as either 'capped' or 'uncapped'. In checkers, for instance, each input connection might represent a square on the board and its value, the piece it contains, while a single output might represent the strength of that position. The game playing agent could then assess the strength of the different positions reachable by its legal moves and choose the best. As an example of an actual checkers program, (Chellapilla & Fogel 2001), the authors used as neural network inputs, sub-divisions of the 8x8 chequers board; in essence they divided it into all possible 3x3 overlapping sub-squares, all possible overlapping 4x4 sub-squares up to the single 8x8 square (the whole board). Each sub-square produces an input that reflects the number of chequers on it. These inputs to the neural network therefore gave it the capacity to learn patterns in the disposition of chequers on the board. For Trap-the-Cap, therefore, this thesis will endeavour to use a form of input connection that allows spatial board patterns to be learnt.

Network output can be represented in one of two ways; either as a single output that gives a value related to the strength of a particular board position, or as a set of outputs that represent the strength of each possible move. As already mentioned, the former is used in conjunction with search algorithms to look ahead and assess different possible moves. This thesis will use the second method. This is the approach taken by Richards et al, (Richards et al. 1998), in their paper on playing the game of Go. The neural network selects a move directly, negating the need for any look-ahead search. The technique relies on the neural network learning to find good patterns in the play. This may or may not be sufficient for a neural network to be a strong player, but it does allow for comparisons between different evolutionary methods to be made.
It has already been mentioned that normally, training data is required to adjust a neural network's weights, this is classified as a supervised learning method. Evolution of a neural network however is a reinforcement learning method; so how does the network gain rewards and hence know when it is improving? This thesis will address this by pursuing the method of competitive co-evolution (Fogel et al. 2004; KO Stanley & R Miikkulainen 2004; Tesauro 2002). The basic idea is to have two populations of game playing agents, initially completely naïve. Select one agent from each population to play each other; play will be more or less random at first since they have yet to evolve any knowledge of strategy. This is repeated for random combinations of agents. The strongest agents are then rewarded by having their genotypes selected to be the parents of the next generation and so on, until one of the neural networks evolves sufficiently strong play.

Competitive co-evolution is simple in concept, but contains many subtleties in its implementation. The first concern is to maintain a diverse population of game playing agents that encompass a range of strategies. If this is not done, then the evolutionary process can 'home in' too quickly onto a single strategy, before having explored the available strategies more fully. The techniques of 'fitness sharing' and 'speciation' can be used to address this potential problem (Sareni & Krahenbuhl 1998). Another pathology that can present itself during co-evolution, is that of repetitively cycling through a set of strategies that are not necessarily very good, but which are good enough for one population to get the better of the other, until that in turn rediscovers an earlier strategy capable of regaining the upper hand. A Co-evolutionary Memory (CM) (Rosin & Belew 1997) represents a suggested means of overcoming this cyclical behaviour. Basically, strong genotypes from earlier generations are retained and used in the assessment of the latest generations fitness; this prevents the evolutionary process from losing earlier successful strategies in the search for new strategies. As an aside, it is interesting that this does not form a part of the evolutionary process in the natural world. If all goes well, then co-evolution can lead to what is termed an 'arms race' (Nolfi & Floreano 1998), in which competing populations reciprocally drive one another to ever higher levels of complexity. This is the ideal situation that co-evolution aspires to.

A criticism of co-evolution is that evolutionary pressure relies entirely on competition with current and ancestral members drawn from the same populations being evolved (Miconi 2009). Miconi states that evolutionary progress can be classified into three types. The first
two are 'local progress' made against current generation opponents and 'historical progress' made against opponents from previous generations. These are what conventionally constitute co-evolutionary algorithms, as has been discussed. There is a third type, global progress that Miconi defines as

“'global progress', (superior performance against the entire opponent space)”,

and he argues, is not necessarily achieved by relying solely on the first two methods of achieving progress. He demonstrates his thesis by using a 'Cross-Validation' method for the measurement of progress in a particular co-evolution experiment. This relies on competition between champion neural networks drawn from each generation and, crucially, a set of opponents, drawn from two or more independently co-evolved populations and which act as a 'test set'. The author states that,

“the confusion between historical and global progress is a case of a common error, namely using the training set [opponents from previous generations] as a test set. This error is prevalent among standard methods for co-evolutionary analysis”

Miconis' Cross-Validation exposed inadequacies with using own population members to measure progress in co-evolution. The author also speculates that some of the undesirable behaviours or pathologies of co-evolution are caused by this lack of an independent test set.

1.3 Aim of Proposed Research

This project will extend the application of Cross-Validation and use it within and as part of the evolutionary process. It is possible to do this by running a Cross-Validation co-evolution experiment and one (or more) other, independent co-evolution experiment(s) side-by-side. The Cross-Validation element will be 'injected' into the evolutionary process at each generation, by taking generation champions from the independent experiment(s) and placing them into the CM of the Cross-Validation experiment. In this way, the fitness of each generation will be partially determined by an independent test set, one that is appropriate to the generation being assessed.

The board game of Trap-the-Cap will provide the testbed for this work, in that the neural networks evolved will learn to play this game. An assessment of the effect of Cross-Validation within the co-evolutionary process can be made by comparing Trap-the-Cap players generated using standard co-evolution and using Cross-Validation. The effect of what proportion of independent Trap-the-Cap players the CM should contain will also be studied.
The co-evolutionary algorithm that will be used is known as NEAT, which is an acronym standing for 'Neuro-Evolution of Augmenting Topologies'. The algorithm was developed by Stanley and Miikkulainen (Stanley & Miikkulainen 2002). There are a number of reasons for choosing this algorithm; firstly, it starts the evolutionary process with a population of neural networks that are minimal in size and which can be added to as evolution proceeds. This increases the speed of evolution as it keeps the amount of computation required to a minimum. Secondly, it successfully incorporates the crossover operation into the genetic algorithm; this has been problematic in prior algorithms for the evolution of neural networks (Angeline et al. 1994; Yao 1999).

### 1.4 Research Question

Does the use of Cross-Validation during the co-evolution of neural networks, provide measurable benefits in terms of speed of evolution, network complexity and performance when compared with standard competitive co-evolution?

What proportion of Cross-Validation CM members should the co-evolutionary process use?

### 1.5 Contribution to Knowledge

This thesis investigates the application of evolutionary techniques to neural networks, for problems where a solution is not known in advance. As technology advances and it becomes possible to address more complex problems, it is imperative to determine the best general approaches that can be applied across a broad range of applications. This will help to move AI away from the bespoke solution towards a situation where engineers not expert in computer science can apply it. This research aims to contribute towards making co-evolution a more stable process, that can more reliably find 'global' solutions for a given problem.

In particular co-evolutionary Cross-Validation will be investigated. Nolfi and Floreano have investigated arms races (Nolfi & Floreano 1998), the ideal that co-evolutionary algorithms aspire to. Their results show that when a new generation judged its fitness with respect to its contemporaries, then problems were prone to occur, in particular the cyclic rediscovery of older strategies. They introduced the 'Hall of Fame' concept in to the fitness assessment process, in which the best individual from every generation is retained for
future testing. This innovation did allow arms races to develop, but did not produce strategies which were much stronger than previously, indeed, some strategies evolved without the Hall of Fame were much superior. A separate strand of work investigated techniques for measuring progress during a co-evolutionary experiment. Examples of popular standard methods of measuring progress are the Master Tournament and the Dominance Tournament. These all work by holding competitions between the newer strategies and older strategies from earlier generations to judge if 'progress' is being made. However, Cross-Validation exposed inadequacies with using own population members to measure progress, in the sense that they do not reliably find the fittest individuals (Miconi & Channon 2006; Miconi 2009). The same criticism could therefore be aimed at the fitness assessment itself, since it is own population members that are used to drive the co-evolutionary process! This thesis will therefore extend Cross-Validation in to the co-evolutionary algorithm itself.

The audience for this work consists of all those who need to deploy applications that cannot realistically have methods of solution designed in advance; either because knowledge to do so is not available or interactions are too 'non-linear'. This will include those involved in manufacturing, communications and distribution among others. Also, there is a large games community, interested purely in the aspects that could be applied to other games and in particular interactive games.
2 Literature Survey

The Introduction outlined a justification for the direction of this thesis based upon the co-evolution of neural networks that can play Trap-the-Cap. In doing so, it gave brief descriptions of genetic algorithms, neural networks, co-evolution and board representation. This section will discuss in more detail the relevant literature; specifically the evolution of neural networks and the important subject of co-evolution and fitness sharing.

2.1 The Reinforcement Learning Method

One of the goals in the study of artificial intelligence is to develop systems requiring zero expert knowledge of the problem domain. This is essential when an agent has to operate in uncharted territory, either because the problem is too complex for there to be any teacher or the problem is interactive and future conditions cannot be predicted.

In their book 'Reinforcement Learning: An Introduction' (Sutton & Barto 1998), the authors outline methods of achieving this ideal. They devised a version of reinforcement learning that assigns an optimum action to every possible state of a system, either explicitly or via approximations to groups of states. The technique relies on numerical awards assigned to known states, usually terminal states, which are propagated back through the state space as the agent explores it. Reinforcement Learning doesn't require the use of neural networks, but in realistic problems, the state space is so large its values must be approximated, and neural networks can perform that task. The first successful application of this method to a non-trivial task was Tesauro and his backgammon playing program TDGammon (Tesauro 1992). He used a fixed architecture neural network as the means to choose which action to take, (the next move), and trained the weights. He used the standard back-propagation method of updating the weights, where the training data of complete games was generated by playing a version of the network against itself. This initial version of TDGammon reached a strong intermediate level of play and was a significant success. Kaelbling et al (Kaelbling et al. 1996) in their survey of Reinforcement Learning stated that in the years following TDGammon, no success close to it was repeated in other games, for example Chess (Thrun 1995). This was partly attributed to the deterministic nature of the other games attempted; they use no die to determine play, unlike backgammon. The fact that backgammon uses die enables more of the state space to be explored. Also, work by Mcdonald et al (Mcdonald & Hingston 1997)
indicated that Reinforcement Learning has potentially severe difficulty in scaling up to large state spaces.

2.2 Evolution of Fixed Neural Network Architectures

Traditionally, neural networks learn using teaching data sets (Gurney 1997) but since complex problems don't necessarily have these, a way was needed to teach them with zero knowledge, just measured outcome. Another avenue of investigation is therefore to apply the principle of Darwinian evolution to the generation of neural networks. Chellapilla and Fogel in their checkers playing program (Chellapilla & Fogel 2001) used this principal to evolve a population of neural networks. Each network evaluated the board position and hence could be used as the basis of a minimax search tree, game playing agent. The fitness of each neural network was assessed by playing it against other, randomly chosen, members of the population. The top neural networks were retained and each became a parent for a child neural network. Each child was generated from its parent by making small random variations to the parents weight; no crossover operators were used. This resulted in a player that was very successful in defeating human players on a checkers playing website (99.61% success rate). Chellapilla and Fogel used a fixed neural network architecture which the authors felt was a reasonable choice based on previous experience. Fogel et al (Fogel et al. 2004) repeated this work using a very similar technique, but using the game of chess instead. Their chess playing program could perform above master level.

The work described so far has relied on fixed architecture neural networks, used because that architecture seemed reasonable. A better approach is to allow the architecture to evolve so that it may be optimised in some way. The next group of methods follow this philosophy.

2.3 Evolution of Neural Network Architectures

Recurrent neural networks were evolved by (Angeline et al. 1994) using the technique of 'evolutionary programming'. Evolutionary programming distinguishes itself from genetic algorithms by using mutation as the sole reproductive operator. Angeline et al argued that the crossover operator is completely inappropriate when applied to neural networks and should not be used. The underlying argument was that the functionality of a network is

---

1. A recurrent network has at least one connection routed from an output or a hidden node to an input of another node that is on the input side of the node in question. Thus a feedback loop is formed which gives the network a short term memory.
distributed across all of the nodes, and no one node can be identified with a single aspect of a network's behaviour. This makes it highly unlikely that an arbitrary crossover operation will produce viable offspring; instead, a mutation operator created offspring within a specific behavioural locus if its parent. The algorithm itself created a population of networks, all having the same pre-defined input and output nodes, but having a randomly defined number of hidden nodes and connections. Each network was assessed for its chosen task and the top 50% selected to produce offspring, the rest being discarded. Offspring production then involved mutation of weights and architectural change with the addition or removal of nodes and links. The results indicated that their method evolved solutions substantially faster than methods relying on crossover operators.

The findings of Angeline et al were reinforced by (Yao 1999) in his extensive review of the evolution of neural networks. He concluded that the simultaneous evolution of connection weights and architectures generally produced better results. He also concluded that crossover operators produce more harm than benefit because they destroy knowledge learned and distributed among different connections. The crossover operator, ordinarily is supposed to swap 'building blocks' of genes to create offspring; but the knowledge in neural nets is distributed over the whole network, with no easy way to find useful groups of genes.

### 2.4 Neuro-Evolution of Augmenting Topologies (NEAT)

Stanley and Miikkulainen, in their paper 'Evolving Neural Networks through Augmenting Topologies' (Stanley & Miikkulainen 2002) introduced a number of innovations that addressed the issue of crossover and improved upon other aspects of previous attempts at evolving neural networks. Called Neuro Evolution of Augmenting Topologies (NEAT), the method starts the evolution process with a uniform population of chromosomes that consist merely of the input and output connection genes. This population is then subjected to the standard genetic algorithm; networks are built from the chromosomes, assessed for performance, parents selected and offspring generated through crossover and mutation. Whenever a child network results from crossover, it undergoes mutation according to some small probability. The mutation operator can make random changes to the network weight values, as is standard practice; it can also add new structure in the form of new connections or new hidden nodes. The structural elaborations that occur are always justified by subsequent fitness evaluations and ultimate survival of the network. This is an important
aspect of the NEAT method since the size of the network starts small and speed of evolution is consequently fast. The key innovation is that whenever a gene is added by the mutation operator (representing a new connection or new node), it is marked with a 'generation number'. The generation number of a gene never changes, when crossover occurs it is copied to the child chromosome so that two genes in different chromosomes, with the same generation number must represent the same structure (possibly with different weights). The crossover operator therefore aligns the selected chromosomes generation numbers, so that swapped genes are guaranteed to represent the same structure. This prevents the distributed nature of the neural networks from being disrupted. An extension of NEAT, called FS-NEAT (Whiteson et al. 2005), automatically determines how best to connect the provided inputs to the neural network; FS is an acronym for Feature Selection. FS-NEAT starts with a network containing no connections at all, and is little more than pools of inputs and outputs. To allow the evolutionary process to get under way, a single connection is added to each network in the population, joining a randomly selected input and output. Evolution proceeds as normal, with mutations adding hidden nodes and connections. Links added from input nodes will be tested by evolution, with only the most useful surviving. The authors claim that;

“FS-NEAT can learn better networks and learn them faster than regular NEAT”.

2.5 Evolutionary Arms Races and Co-evolution

The concept of an evolutionary 'arms race' was discussed by Dawkins and Krebs, (Dawkins & Krebs 1979), for the biological sciences. They suggested that,

“An adaptation in one lineage (e.g. predators) may change the selection pressure on another lineage (e.g. prey), giving rise to a counter-adaptation. If this occurs reciprocally, an unstable runaway escalation or 'arms race' may result.”

This idea was applied to evolutionary algorithms and gave rise to competitive co-evolution and the related subjects of fitness sharing, speciation and co-evolutionary memories.

Competitive co-evolution is an evolutionary technique that can learn to perform a task, without the aid of problem specific knowledge. An example of such a system, which later formed the basis of a NEAT co-evolutionary algorithm, is the work of Rosin and Belew (Rosin & Belew 1995). They defined competitive coevolution as the

“Simultaneous evolution of two or more genetically distinct populations with coupled fitness landscapes”.

13
Rosin and Belew's method consists of two elements, referred to respectively as 'competitive fitness sharing' and 'shared sampling'. The population of neural networks is split into two separate groups which take it in turns to assess the others' fitness. Competitive fitness sharing assigns higher fitness scores to networks that can defeat networks from the other group that few others can. This might occur due to a network evolving a new innovation, or an old one that has recently become important for instance. Hence, the survival of important, but poorly represented individuals is enhanced and this helps to maintain a greater number of diverse 'niches' within the population. Shared sampling saves computational effort and increases speed; when one group of networks is acting as the tester it is usual to select a subset of networks for the task. The method used to select this subset that will challenge all segments of the group being tested, is to simply choose networks that have high competitive fitness sharing scores, determined from their previous rôle as the group being tested. Competitive co-evolution ideally sets up an evolutionary 'arms race' with the two populations competing at a similar level of ability, but taking it in turns to evolve better strategies than the other.

### 2.6 Fitness Sharing

Fitness sharing in its various forms is an important part of the co-evolutionary algorithm. An introduction to the subject is given by several authors (Miller & Shaw 1996; Della Cioppa et al. 2007). The problem to be solved by co-evolution was viewed as a 'fitness landscape' that had a number of peaks distributed across it. Each peak represents some type of optimal behaviour. In standard evolutionary algorithms, the fitness landscape does not change and there is a tendency for the selection mechanism to converge on a population of very closely related solutions close to one peak in the fitness landscape. In contrast, in co-evolution, the continual interplay between the two populations causes the fitness landscape to be constantly changing. It is therefore important to ensure a broad range of solutions exist within both populations so as to respond to new challenges as they evolve. 'Crowding' was an early method of achieving this, it basically chose members of a population to reproduce and the resultant offspring would only replace population members that were similar to those offspring. This slowed down the tendency to converge onto a single solution, but retarded exploration of solutions not in the initial population. Fitness sharing was introduced by Holland (Holland 1975) and improved upon by Goldberg and Richardson (Goldberg & Richardson 1987), who developed the method of 'Explicit Fitness Sharing'. Since the location of the fitness peaks was not known, it was the
job of the co-evolutionary algorithm to find them. A subset of an evolved population that displayed similar properties was interpreted as occupying one of these fitness peaks. To encourage all of the peaks to be discovered, the fitness of each population member was reduced, or 'shared', depending upon how many other members of the population it was similar to. The greater the number of similar individuals the greater the reduction in fitness. This allowed population members with few similar individuals to have an enhanced fitness score and hence a chance to see if they were near to a peak and increase in number. An analogy is drawn with natural ecosystems, where different species occupy different niches in the environment and an environmental niche is identified with a fitness peak. The number of representatives of a species that can survive within a niche depends on the resources, e.g. food and water, that are available to them. Fitness is viewed as a resource which becomes more scarce as a fitness peak becomes crowded.

The analogy with natural ecosystems can be taken further by explicitly defining a group of similar population members in the region of a fitness peak, to be a distinct species. The definition of a species in this sense is that under reproduction, members of a species may only reproduce with members of the same species. Reproduction between species is deemed to have a high chance of producing poorly performing offspring. This was not a part of the original Explicit Fitness Sharing method, but was incorporated into a method used for standard evolutionary algorithms called 'Dynamic Fitness Sharing' (Della Cioppa et al. 2007). In standard artificial evolution, the fitness landscape is fixed and it just remains for all of the peaks to be discovered and populated by species. The situation is subtly different when fitness sharing and speciation are applied to NEAT (Stanley & Miikkulainen 2002). Firstly, speciation in NEAT is easily achieved by comparing the list of innovation numbers of the genes in a genome with those of another genome. The number of differences dictates how closely related the two genomes are. A 'compatibility threshold' determines if the two genomes are close enough together to be considered members of the same species. After reproduction, it is possible that new genomes will become sufficiently different to warrant the creation of new species, this also occurs in standard evolution methods. However, in NEAT new structure can be added during reproduction causing the fitness landscape to be altered. In this new fitness landscape, a new species may be superior to an older one; or to look at it another way, the new species occupies a significant fitness peak, whilst the old species sees its fitness peak become relatively much smaller. This then could lead to species extinction. During reproduction
in NEAT, every species is assigned a number of offspring in proportion to the overall fitness of that species. Then the best-performing r%, where r is a fixed quantity, of each species is randomly mated to replace the entire population of that species. Perhaps the most important aspect of speciation for NEAT, is the protection of innovation. When a chromosome mutates and acquires new links or nodes, its real fitness may be reduced; however, since the new mutation is competing only within its species, it is competing within its own niche and its shared fitness protects the structural innovation long enough to optimize.

2.7 The Hall of Fame and Co-evolutionary Memories

The concept of an arms race, or two populations competing to evolve better strategies, has already been briefly introduced. Nolfi and Floreano (Nolfi & Floreano 1998) asked the question “Do arms races arise in artificial evolution”; the answer was not as clear cut as the simple proposition of ever better strategies suggests. They found that the continuous increase in complexity of strategy was not guaranteed and that populations could in fact cycle between strategies. These cyclical temporary improvements over the co-evolving population meant there was no incentive to discover new strategies. They also found that individuals of later generations did not always score well against earlier generations. This was ascribed to effective strategies being 'lost' during evolution. Similar behaviours probably exist within natural evolution, but are not acceptable in artificial evolution since the aim is to achieve as close to an optimum solution as possible. Rosin and Belew (Rosin & Belew 1997) introduced the Hall of Fame as a possible remedy.

The Hall of Fame is a member of a class of methods known collectively as Co-Evolutionary Memory (CM) which aim to retain valuable individuals from previous generations. Future generations then use the CM as part of the fitness assessment process, thus preventing them from forgetting previous strategies. Despite not being an optimal CM (Monroy et al. 2006), the Hall of Fame is the method of choice for competitive co-evolution due to its computational simplicity. It simply stores the generation champion of every generation as an archive to use for fitness testing. There is no mechanism for identifying and removing members of the archive that may no longer be relevant.

A subject related to CM, in that it encourages an arms race and the retention of prior strategies, is complexification. Complexification in evolutionary computation refers to
adding structure, for example new connections or nodes to a neural network, without losing the existing capabilities. In this way, new more complex strategies elaborate on simpler ones and are also likely to maintain existing capabilities. In NEAT, complexification is realised through the mutation operator (Stanley & Miikkulainen 2004). Stanley and Miikkulainen point out that with a fixed genome size, when a good strategy is found, the entire genome is used to encode it. The only way to improve the strategy is to change it and this may contribute to the cyclic rediscovery of solutions in co-evolution. In principal, complexification allows coevolution to continue improving strategy indefinitely by continually 'seeding' the arms race between the two coevolving populations. It is interesting that the authors continued to use a Hall of Fame in their experiments to maintain prior capabilities, rather than trust to complexification to do the job on its own. Stanley and Miikkulainen concluded that when compared with the evolution of fixed networks, complexification discovers significantly more sophisticated strategies.

2.8 Cross-Validation and the Measurement of Progress

The method of Cross-Validation (Miconi & Channon 2006) for measuring evolutionary progress has already been introduced. The authors demonstrated that Cross-Validation can find the best strategies more effectively than methods relying on test strategies drawn from the training set. Ancestral opponents are the co-evolutionary version of a training set. Miconi (Miconi 2009) discusses this in terms of superiority and progress, where he defines progress as being equivalent to superiority over ancestors. In measuring superiority the author states that;

“the superiority of an individual A over an individual B can only be determined... by comparing performances against a common set of opponents”

Progress can be measured against three types of common test set.

<table>
<thead>
<tr>
<th>Progress Type</th>
<th>Definition</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local progress</td>
<td>'A' is better than 'B' (current opponents)</td>
<td>Effectively using training set to measure progress.</td>
</tr>
<tr>
<td>Historical progress</td>
<td>'A' is better than 'B' (current &amp; ancestor opponents)</td>
<td>A CM (archive) makes historical progress identical to local progress.</td>
</tr>
<tr>
<td>Global progress</td>
<td>'A' is better than 'B' (all possible opponents)</td>
<td>Requires knowledge of unknown opponents to form part of the test set.</td>
</tr>
</tbody>
</table>

Table 1: Types of co-evolutionary progress
Measuring progress against a training set is not usually recommended in machine learning applications, since it raises the risk of 'over fitting' where individuals are strong against the training set, but poor against unseen individuals. Miconi demonstrates the superiority of Cross-Validation as a means of the analysis and evaluation of competitive co-evolution; he only implies using Cross-Validation to provide stronger evaluation of fitness during co-evolution when discussing 'over fitting'. At least some opponents used for evaluating fitness should be evolved independently of individuals being evaluated to give a truer indication of global progress.
3 Research Methods

This section will discuss the research methods to be used in this thesis. It will start by drawing together and summarising the co-evolutionary framework outlined in chapters 1 and 2. Then, a summary of evaluative measurement techniques for speed of evolution, complexity and performance will be given along with the final choice of the techniques to be used. Finally, the application of these techniques to Cross Validation assessment will be discussed.

3.1 Project Framework

The game of Trap-the-Cap has no repository of knowledge specifying optimum play or recognised champion players, either human or machine. The Trap-the-Cap player developed must therefore, by necessity, discover strong play by itself. This implies using the class of methods that come under the general approach of reinforcement learning; but which method to choose? This is a free choice, governed purely by finding an interesting and useful question to answer within one of the possible techniques. The general method chosen for this thesis is that of genetic algorithms (Mitchell 1998), which rely on evolutionary pressure to provide feedback and reinforce desired behaviour; while the actual game playing agent will be implemented as a neural network. This choice was inspired by the work of Fogel and Chellapilla into the evolution of checkers and chess playing, neural network agents (K. Chellapilla & D.B. Fogel 1999; K. Chellapilla & D.B. Fogel 2001; D.B. Fogel et al. 2004). Ordinarily, neural networks use supervised learning techniques to acquire knowledge of the problem; however, when encapsulated within a genetic algorithm they fall within the domain of reinforcement learning.

One aspect of genetic algorithms that must normally be considered is the requirement for a fitness function. It is the fitness function that would assess each chromosome for its ability to produce a strong Trap-the-Cap player. Since the definition of a strong player is not known, the need for an explicit fitness function is dispensed with and replaced by Competitive Co-evolution, where two separate populations of chromosomes take it in turns to quantify the fitness of the other. The fitness function is therefore replaced by a simple count of the number of matches won by each neural network in a population when playing the other population. The two populations start out as naïve (random) players, but gradually improve as the best are selected for reproduction. The competitive co-evolution
method will follow that of Rosin & Belew (Rosin & Belew 1995). Two separate populations of neural nets will be maintained and evolved, with no genetic material being exchanged between the groups. Each group will take a turn in being tested by the other group for fitness. There are three elements to the method:

a) Competitive fitness sharing: An individual's fitness is divided by the number of 'similar' individuals in the population. This rewards unusual individuals that might not be the fittest, but potentially have useful behaviours.

b) Shared sampling: Reduce the number of games that have to be played by the group being tested by only selecting a subset from the testing group. Use the competitive fitness score from the group that has just been tested and is now providing the testing to select individuals for competitive matches. This selects those individuals that few members of the current test group did well against.

c) Hall of fame: Saving individuals that have done particularly well to continue contributing successful genetic material and to continue to provide a challenging test set.

Game playing neural network agents, can have their outputs configured in one of two general ways. Firstly, a single output can give an indication of the strength of a specific situation, allowing search algorithms to discover the best of the legal ways forward. However, since search algorithms can introduce significant computational overhead, the second method was chosen for this thesis; a multitude of outputs, each one representing the strength of one of the available legal moves. The highest of the output values is simply chosen as the next move. This may result in players that could be stronger; for example, Tesauro (Tesauro 2002) found that he could increase the strength of play of his neural network Backgammon agents by introducing some look ahead search. However, since the objective of this work is comparative rather than finding champion players, the reduced computational overhead route is preferable.

To summarise then, competitively co-evolved neural networks with multiple outputs, each representing the strength of a legal move, will be used. The question remains of how to implement the neural network; should it have fixed architecture and which genetic operators should be used? The chosen method was 'Neuro-Evolution of Augmenting...
3.2 Evaluative Measurements

To make comparisons between co-evolutionary methods, it is necessary to quantify aspects of their behaviour. The principal metrics chosen are:

1) Speed of Evolution
2) Network Complexity
3) Network Performance.

These then need to be defined and methods selected for their measurement.

Measurement of absolute progress during competitive co-evolution is problematic in that there is no external measure available, all fitness comparisons are done relative to current or ancestral members of the populations; indeed, this is why co-evolution is often selected in the first place when there are no external teachers available. This is sometimes referred to as the 'Red Queen effect' (Cliff & Miller 1995). The Red Queen was a living chess piece in Lewis Carroll's 'Through the Looking Glass', she ran perpetually without getting very far because the landscape kept up with her. By analogy, the fitness landscape continually changes as the two populations evolve. There are a small number of methods described in the literature capable of quantifying the various aspects of competitive co-evolution and which are computationally tractable.

The “Dominance Tournament” (Kenneth O Stanley & Risto Miikkulainen 2002) attempts to find 'significant' game playing agents as the populations evolve. The method works as follows: In each generation, the fitness of every member of each population is determined for the purposes of reproduction; using this data, the best performing member from each of the two populations can compete to find the generation champion. It is from the generation champions that so called dominant strategies are identified. The first dominant strategy is simply defined as the generation champion of the first generation. In all subsequent generations, the latest generation champion plays all previous dominant strategies until it
loses a match. If it never loses a match, then it becomes the latest dominant strategy. The generations at which new dominant strategies emerge indicates that the arms race is still progressing; when no new dominant strategies appear, the arms race has stagnated. This is then a method which can be used to compare the relative 'speed' of evolution of two different co-evolutionary experiments.

The “Best of Run” comparison (Monroy et al. 2006) starts by finding the single best player from each co-evolutionary experiment being compared; for a given co-evolutionary experiment this is the winner of a Master Tournament (Floreano & Nolfi 1997) among the fittest players of every generation and represents the solution of the co-evolution, the real output. The Master Tournament winners from different experiments are then compared by competition to determine which experiment used the better co-evolutionary method. A Master Tournament consists of the champion from each generation being compared with the champions of all other generations and the strategy that defeats the most other champions is declared the winner. This is unfortunately a very time consuming method and requires fewer games to be played between each pair of generation champions in order for it to complete in a realistic time.

The “Best of Generation” comparison (Monroy et al. 2006) attempts to approximate an absolute fitness test which can be used to measure progress in a co-evolutionary experiment, generation by generation. For all of the co-evolutionary methods being compared, the Best of Generation individuals from each are collected together to form an evaluation set. This evaluation set is used to determine the fitness of every generation champion for each experiment, giving the required generation by generation profile of progress for each experiment. This analysis can only be performed after the experiments have finished. It is interesting that, although Monroy et al do not mention it, this would in fact be a cross-validation method if the evaluation set used is drawn from players generated independently of the players being tested.

The “Equal Effort” comparison provides similar results to the Best of Generation comparison, except that it is a relative measure of progress between two co-evolutionary methods. At a chosen generation, all current generation champions from one method play all the generation champions of the other method and the respective number of victories is recorded. This allows the relative speed of evolution for the two methods to be compared.
Network complexity can be easily determined by counting the number of genes in each chromosome. A summary of the comparison methods is given in the following table.

<table>
<thead>
<tr>
<th>Method</th>
<th>Parameter</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dominance Tournament</td>
<td>Speed of evolution</td>
<td>Measures generation at which significant players appear for a single co-evolutionary experiment. Can be used to compare with other experiments.</td>
</tr>
<tr>
<td>Best of Run</td>
<td>Network performance</td>
<td>Finds the best player in a co-evolutionary experiment. Can be used to compare with other experiments.</td>
</tr>
<tr>
<td>Best of Generation</td>
<td>Speed of evolution</td>
<td>Approximates an absolute fitness assessment using two or more co-evolutionary experiments.</td>
</tr>
<tr>
<td>Equal Effort</td>
<td>Speed of evolution</td>
<td>Relative comparison between two co-evolutionary experiments.</td>
</tr>
</tbody>
</table>

Table 2: Comparison Method Summary

The techniques chosen for this thesis are:


The network complexity and network performance techniques were an easy choice. The speed of evolution measurement technique had more contenders. The Dominance Tournament was chosen because it has already been applied to the NEAT co-evolutionary method and has been shown to be superior to its competitors by highlighting evolutionary stagnation, as well as requiring fewer games to be played to generate its results.

### 3.3 Cross Validation Assessment

The central question addressed in this thesis is based upon a criticism of co-evolution (Miconi 2009). The criticism is that training and test opponents in competitive co-evolution are effectively one and the same thing. This is traditionally thought of as a bad
thing in AI, since the evolved agent will not necessarily develop into a strong player capable of defeating previously unseen opponents and may also lead to overly optimistic measurements of its evolutionary progress.

Miconi applies Cross-Validation to the measurement of evolutionary progress, by using independently generated agents as his 'yardstick'. This thesis investigates the effects of using cross-validation as part of the evolutionary process itself. It is therefore necessary to compare co-evolution with and without Cross-Validation.

Cross-Validation will be achieved by adding an extra Co-evolutionary Memory (CM) into the co-evolution of two populations. This extra CM will provide test agents, in the same way that test agents are drawn from the normal CM, (generation champions) and from the current tester population. The members of this extra, or Cross-Validation CM will contain generation champions, generated from previous co-evolutionary experiments, completely unrelated to the co-evolution of populations in the current experiment. At each generation, the competitors provided by this independent test set will be selected such that they will have undergone the same or fewer generations of evolution than the population members of the co-evolution experiment.

The Dominance Tournament, node/link counts and the Best of Run measurement methods, will provide the means of drawing comparisons between co-evolution experiments undertaken with and without Cross-Validation.

3.4 Implementation

A brief overview of the software design will be given, along with the description of a few concepts which are required for understanding the results.

The implementation of the software was based upon the C++ source code of Stanley, (K Stanley 2001) for the NEAT neural network architecture. It was written in Java and adds the ability to use an extra Co-evolutionary Memory as a source of Cross-Validation neural networks. The high level organisation of the software is shown in Error: Reference source not found. There is a conceptual layer, where two populations of organisms evolve by playing games between the organisms of each population to assess organism fitness. The actual game played and the actual type of organism evolved, (in this case Trap-the-Cap and
neural networks respectively), can be plugged into the conceptual layer.

The Coevolution class:

This controls the co-evolution process; it has two CoevolutionaryMemory objects, one for storing generation champions generated during co-evolution and one for storing generation champions from a completely separate and unrelated coevolution run. The former is the

Figure 2: High level class diagram of Co-evolution software.

The Coevolution class:

This controls the co-evolution process; it has two CoevolutionaryMemory objects, one for storing generation champions generated during co-evolution and one for storing generation champions from a completely separate and unrelated coevolution run. The former is the
'Hall of Fame' and the latter provides networks for Cross-Validation. The Coevolution class also has the two Population objects which co-evolve.

The Population class:
This keeps a collection of all the Network objects and Species objects which form the population. The population of networks is created from a single 'seed' network, which is mutated to generate each network required by the population. The networks are each assigned to a Species object, (only one for each Network). Each species is designated as being different from all the other species as determined by topological similarity, known as compatibility.

**Core Algorithm for Co-evolution**

The following runGenerations() method is performed for each of the two populations in turn, with each population taking it in turns to be the 'hosts' or the 'testers'.

- **Coevolution.runGenerations()**
  - **Coevolution.fitnessEvaluation(hosts, testers)**
    - Play each 'host' against every 'tester' and record the number of wins for each host. This represents the fitness score for each host.
  - **CoevolutionaryMemory.addOrganism(champion)**
    - Add the host with the highest fitness score to the co-evolutionary memory.
  - **Population.epoch()**
    - Process each host's fitness score based upon Competitive Fitness Sharing.
    - Reproduce, using processed fitness scores to decide the number of organisms that each species will provide into the new population.
    - Assign each organism in the new population to a species.

*Figure 3: Outline of core co-evolution algorithm.*

The central algorithm is performed by the runGenerations() method, (Figure 3), in the
Coevolution object. At the end of every generation each population will have been completely replaced by a new population through reproduction. Each generation is split into two parts; one population has its fitness assessed by playing members of the other population, fitness sharing then modifies the assessed fitness scores which in turn are used to assess how many organisms each species will provide into the new population. This new population is then divided up into species, completing its evolution for this generation. Once reproduction has taken place for the population that has had its fitness assessed, the two populations switch roles and the process is repeated to complete one generation of evolution.

The process of splitting up a population into species is governed by the concept of 'Compatibility'. The compatibility of two networks, is the sense of how similar two networks are to each other in terms of behaviour. This is measured by determining which links they have in common, and which links they do not. The sum of the weight differences of similar links and the sum of weights of distinct links are added. The resultant number expresses the degree of similarity. A value of zero represents identical behaviour, even if the link structure is different. A larger number represents increasing dissimilarity in behaviour. The 'compatibility' value lies in the range 0 to a positive value. The compatibility is defined in equation 1.

\[
e = m \cdot C_1 + d \cdot C_2 + e \cdot C_3
\]  

(1)

Where \( m \) is the sum of matching link weight difference magnitudes, \( d \) is the sum of disjoint link weight magnitudes and \( e \) is the similar value for excess links. The definitions of disjoint and excess links are shown in Figure 4. The figure shows the process of merging the links from two networks when mating. Each rectangle represents a link, the number in the top of each rectangle is the identifier for that link. Links with the same identifier are connected to the equivalent nodes in both networks. If a link is present in one network but not the other then it is defined as either being disjoint or excess. When a network comes to be placed into a species, it is compared against the fittest network from each of the species in the previous generation in turn. Each comparison consist of calculating the compatibility between the network and the example network from the previous generation. If the compatibility is below a threshold value, then it is placed in that species and it undergoes no further comparisons. This continues until all of the networks have been
placed into a species. If a network finds no species that it is compatible with, then a new species is created.

Each neural network was supplied with 133 input values and provided 14 output values. The first 132 input values represent the state of the Trap-the-Cap board. Each cap position is represented by a value:

a) +ve for current player caps (the player whose move it is).

b) -ve for all other players.

c) 0 for empty spaces.

The number of spaces on a Trap-the-Cap board is 73, however, when the safe spaces are include three times (once for each potential occupant) and the home bases are included 6 times for the same reason, the total is 127. Then when the 'Hub' value is repeated six times, once at the end of each spoke, the total number of values is 132. The value assigned
to each possible cap position is \( \pm (1 + \text{number of capped caps}) \). So the magnitude of the input value represents the number of 'capped' caps on a space and its sign determines if the occupying cap belongs to the player whose turn it is. The final 133\(^{rd}\) input to the neural network is given the value of the die throw.

The 14 output values are split into two groups of 6 and 8. The first group of 6 outputs represents each cap a player owns and the highest value of the 6 outputs determines which cap to move. If a player has had caps captured, then only the first \( n \) outputs are examined, where \( n \) is the number of caps the player has available to play. The second group of 8 outputs determines which move to make. If a cap can move, then there are at most 8 possible legal positions to move it to and at minimum 1. Once a given cap has been selected, the possible legal moves are determined and ordered according to a hierarchy of move type:

1st) Move capped cap back to home base.
2nd) Move cap to cap another player.
3rd) Move cap to a safe space.
4th) Move cap to ordinary, empty space.

If \( m \) moves are available, then the largest value of the first \( m \) values of the 8 outputs determines the move to select. The possible moves are put in order to provide consistency for the neural network; the actual ordering is unimportant.

If co-evolution is to take place using Cross-Validation, then a file containing generation champions from a completely independent co-evolution experiment is provided as a co-evolutionary memory. If this is not provided, then ordinary co-evolution takes place.
4 Experiments

This section will discuss some of the parameters used in the experiments, the co-evolution experiments that were performed and finally, the type of data recorded.

4.1 Introduction

The software was initially verified by testing each class method with data whose results could be calculated by hand. It was not possible to verify the complete software system against a hand calculated test as this was far too complex. Instead, the co-evolutionary process was observed, to determine if its behaviour conformed to expectations; this will be discussed in the next chapter. The software was run on Intel Core 2, 64 bit processors and due to the size of the neural networks involved and the number of games that had to be played, each experiment took no less than five days and in some cases up to seven days to complete. As much tuning of parameters as possible was performed using small population sizes and few generations, however, the final tuning of parameters had to be completed using actual experimental values, which of course consumed a lot of time. The parameters themselves will be discussed in the next section. The net effect of this is that, although the final experiments produced convincing behaviour, it was not realistic to draw statistical inferences from them.

4.2 Parameters used for Evolution

An example of the parameter values given to a co-evolution experiment is shown in Figure 5. They will not all be discussed here; as far as possible, the values were set to those chosen by Stanley (Stanley 2001), (Stanley & Miikkulainen 2004, Appendix A). Some of the more significant parameter choices are however discussed below.

The number of players was set at two for each game, this was to reduce the time taken since increasing the number of players (up to the maximum of 6) increased the number of moves to make. This was important since the number of games required to be played was approximately 2.9 million for a single experiment consisting of 500 generations as well as 0.5 million games for a Dominance Tournament and 1.25 million games for a Master Tournament. Having only two players also made it simpler to interpret how games were played.
When a game was played it was necessary to specify a maximum number of moves that each player could make before the game was declared a draw and was initially set to a maximum of 240 moves for each player. This value was chosen to make sure that in the early stages of evolution, when games were essentially random walks around the board, there would be a reasonable number of games that produced a winner. It was discovered after the first set of experiments that better results could be obtained by reducing this value.

---

**Figure 5: Typical Set of Experimental Parameters**

<table>
<thead>
<tr>
<th>Experimental Parameters for Co-evolution</th>
<th>Evolution Information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Game Initialisation Information</strong></td>
<td></td>
</tr>
<tr>
<td>GAME_TYPE = TrapTheCap</td>
<td>DISJOINT_COEFF = 1.0</td>
</tr>
<tr>
<td>NO_PLAYERS = 2</td>
<td>EXCESS_COEFF = 1.0</td>
</tr>
<tr>
<td>MAX_NUMBER_OF_TURNS = 240</td>
<td>WEIGHT_DIFF_COEFF = 0.25</td>
</tr>
<tr>
<td><strong>Co-evolution Initialisation Information</strong></td>
<td></td>
</tr>
<tr>
<td>NO_RUNS = 1</td>
<td>POWER = 1.5</td>
</tr>
<tr>
<td>NO_PARASITES = 4</td>
<td>RATE = 1.0</td>
</tr>
<tr>
<td>NO_GENERATIONS = 4</td>
<td>COMPAT_THRESHOLD = 20.0</td>
</tr>
<tr>
<td>NO_GENERATION_CHAMPS = 4</td>
<td></td>
</tr>
<tr>
<td>NO_CV_GENERATION_CHAMPS = 4</td>
<td></td>
</tr>
<tr>
<td><strong>Population Initialisation Information</strong></td>
<td></td>
</tr>
<tr>
<td>ORGANISM_TYPE = Network</td>
<td></td>
</tr>
<tr>
<td>POP_SIZE = 240</td>
<td></td>
</tr>
<tr>
<td><strong>Network Initialisation Information</strong></td>
<td></td>
</tr>
<tr>
<td>HIDDEN_NODES = 0</td>
<td></td>
</tr>
<tr>
<td>MAX_HIDDEN_NODES = 2400</td>
<td></td>
</tr>
<tr>
<td>RECURRENT = true</td>
<td></td>
</tr>
<tr>
<td>WEIGHT_HARD_LIMIT = 2.0</td>
<td></td>
</tr>
<tr>
<td>LINK_PROBABILITY = 0.0</td>
<td></td>
</tr>
<tr>
<td><strong>Species Ageing and Survival Information</strong></td>
<td></td>
</tr>
<tr>
<td>DROP_OFF_AGE = 50</td>
<td></td>
</tr>
<tr>
<td>STAGNATION_FITNESS_MULTIPLIER = 0.01</td>
<td></td>
</tr>
<tr>
<td>MAX_AGE_OF_FITNESS_BOOST = 15</td>
<td></td>
</tr>
<tr>
<td>AGE_SIGNIFICANCE_MULTIPLIER = 2.0</td>
<td></td>
</tr>
<tr>
<td>SURVIVAL_THRESHOLD = 0.8</td>
<td></td>
</tr>
<tr>
<td><strong>Node Initialisation Information</strong></td>
<td></td>
</tr>
<tr>
<td>SQUASHING_FUNCTION = SIGMOID</td>
<td></td>
</tr>
</tbody>
</table>

When a game was played it was necessary to specify a maximum number of moves that each player could make before the game was declared a draw and was initially set to a maximum of 240 moves for each player. This value was chosen to make sure that in the early stages of evolution, when games were essentially random walks around the board, there would be a reasonable number of games that produced a winner. It was discovered after the first set of experiments that better results could be obtained by reducing this value.
somewhat. This had the added benefit of reducing the run time of an experiment proportionately. Later values used were 120 and 90 moves per player.

Each population took it in turns to have its fitness assessed by playing games against neural networks drawn from the other population and from a Co-evolutionary Memory (CM) of generation champions. Stanley provided 12 networks as testers; 4 from the other population and 8 drawn randomly from the CM. These values were not altered for the experiments in this thesis. The only difference occurred when a second CM was used as a source of neural networks for Cross-Validation as well as the standard CM. In this case, the 8 networks supplied from the CM's were shared to give a total of 8.

In the 'Network Initialisation Information' section of the parameter values there is a value called 'LINK_PROBABILITY', which was always set to zero. This, as it suggests, means that when the seed network for a population was generated it had zero probability of having any links created. Once a seed network was generated it was examined to make sure that every input node had at least one link connected to it, if not one was generated randomly, connecting it to another valid node, i.e not another input node. Also, the number of initial hidden nodes was always set to zero. The result of this is that all populations were created from a minimal network consisting of just one link connected from each input to a randomly chosen output node.

During early testing of the software it was discovered that a small number of the weight values associated with network links took on very large values and distorted the assignment of networks to species. In order to combat this the link weights were hard-limited to a maximum value. The work of Woodland (Woodland 1989) examined hard limiting weight values in multi-layer neural networks and concluded that small values worked well and even improved the generalisation of a network. The experiments in this thesis used the value of +/- 2.0 as recommended by Woodland. With hard limiting in place it was seen in further testing that less than 0.5% of the links took on the maximum magnitude during evolution.

4.3 Trap-the-Cap Experiments

The experiments performed are summarised in Table 3 and Table 4 which also show the most important parameter values associated with each experiment, i.e those values that
distinguish the experiments apart.

### 240 moves per player per game before draw is declared

<table>
<thead>
<tr>
<th>Expt No</th>
<th>Type of Tester Network</th>
<th>N° of Networks</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Parasites</td>
<td>4</td>
<td>Ordinary co-evolution, no Cross-Validation.</td>
</tr>
<tr>
<td></td>
<td>CM</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cross-Validation CM</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Parasites</td>
<td>4</td>
<td>Ordinary co-evolution, no Cross-Validation.</td>
</tr>
<tr>
<td></td>
<td>CM</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cross-Validation CM</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Parasites</td>
<td>4</td>
<td>Cross-Validation with 2 networks.</td>
</tr>
<tr>
<td></td>
<td>CM</td>
<td>6</td>
<td>Expt 1 used as source of networks.</td>
</tr>
<tr>
<td></td>
<td>Cross-Validation CM</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Parasites</td>
<td>4</td>
<td>Cross-Validation with 4 networks.</td>
</tr>
<tr>
<td></td>
<td>CM</td>
<td>4</td>
<td>Expt 1 used as source of networks.</td>
</tr>
<tr>
<td></td>
<td>Cross-Validation CM</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Parasites</td>
<td>4</td>
<td>Cross-Validation with 6 networks.</td>
</tr>
<tr>
<td></td>
<td>CM</td>
<td>2</td>
<td><strong>Expt 1</strong> used as source of networks.</td>
</tr>
<tr>
<td></td>
<td>Cross-Validation CM</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

**Table 3: Experiments with up to 240 moves per player**

### 120 moves per player per game before draw is declared

<table>
<thead>
<tr>
<th>Expt No</th>
<th>Type of Tester Network</th>
<th>N° of Networks</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Parasites</td>
<td>4</td>
<td>Ordinary co-evolution, no Cross-Validation.</td>
</tr>
<tr>
<td></td>
<td>CM</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cross-Validation CM</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Parasites</td>
<td>4</td>
<td>Ordinary co-evolution, no Cross-Validation.</td>
</tr>
<tr>
<td></td>
<td>CM</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cross-Validation CM</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Parasites</td>
<td>4</td>
<td>Cross-Validation with 4 networks.</td>
</tr>
<tr>
<td></td>
<td>CM</td>
<td>4</td>
<td><strong>Expt 6</strong> used as source of networks</td>
</tr>
<tr>
<td></td>
<td>Cross-Validation CM</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

**Table 4: Experiments with up to 120 moves per player**

In Table 3 there are two experiments that used ordinary co-evolution alone (1 and 2), without Cross-Validation. One of these experiments (1) provided the generation champion.
networks for the Cross-Validation experiments (3, 4 and 5). Experiments 3, 4 and 5 all use Cross-Validation during their evolution, but with a varying proportion of networks between the standard co-evolutionary memory CM and the one used as the source of independent Cross-Validation networks. In Table 4 the experiments of Table 3 were repeated but using only 120 moves before a draw was declared in games. Also, only one Cross-Validation experiment was performed.
5 Analysis of Results

5.1 Co-evolution

The first step in the analysis of the experimental results is to demonstrate that co-evolution is occurring between two populations. This was achieved by examining the Best of Generation competitions carried out each generation, the Dominant Strategies and the evolved networks compatibility distance from the seed network.

Generation Champions:
Every generation, the two Best of Generation networks, one from each population, played each other to establish which was the champion. The competition consisted of the network from 'population A' playing the network from 'population B' 100 times. The number of wins achieved by population A was recorded. The rolls were then reversed and the number of times population B defeated population A over 100 games was recorded. A graph of the scores achieved by populations A and B over the 500 generations of evolution is shown in Figure 6. It is not really possible to see what is going on in this graph, so an expanded section of the same graph is shown in Figure 7.

![Experiment 2: Best of Generation scores for each generation](image)

*Figure 6: Best of Generation scores for populations A and B.*
It can be seen from the second graph that there is an interplay between populations A and B; in fact they cycle between being the most successful population. This can be explained as follows. If, for instance, population A is the strongest, then all of its networks will find it easier to obtain a high fitness score during one generation of evolution; however, population B network's will find it more difficult and only the very best of them will achieve higher fitness scores. So the weaker population experiences stronger selection and improves to become the more successful population and so on. It should be remembered that the performance of one population with respect to the other is relative; a population that becomes weaker than its co-evolving partner population could still be improving globally, i.e. becoming a better Trap-the-Cap player.

Another example shows the first few generations of experiment 4, Figure 8, where population B started out with particularly poorly performing networks and was defeated by a wide margin. However, by the 9th generation, it had recovered and the interplay between populations was established.

Figure 7: Best of Generation scores for populations A and B (Expanded scale)
Dominant strategies:
The generations at which Dominant Strategies appeared for all 8 experiments are shown in Figure 9.

Figure 8: A poor start for populations B.

Figure 9: Dominant Strategies for all 8 experiments.
Remember that a dominant strategy is a generation champion that can defeat all earlier dominant strategies. The set of dominant strategies for a co-evolution experiment therefore represents a sub-set of generation champions that form a strictly ordered list of improving performance (if network \(a\) can beat network \(b\) and network \(b\) can beat network \(c\), then \(a\) is guaranteed to beat \(c\) as well). Stanley and Miikkulainen in their paper introducing the dominance tournament (Stanley & Miikkulainen 2002) state that:

“With such a guarantee, the dominance tournament provides proof of progress”

They go on to say that even if an earlier strategy can beat a later dominant strategy, it simply indicates the existence of an idiosyncratic strategy capable of beating a specific dominant strategy. In other words, dominant strategies are the best all rounders and represent a genuine increment in progress.

The emergence of dominant strategies as depicted in Figure 9 therefore indicates that the experiments on the co-evolution of Trap-the-Cap players succeeded in producing evolutionary progress. Looking in more detail and starting with experiments 1 to 5, it can be seen that although all 5 experiments made evolutionary progress, only experiments 1 and 2 maintained this across nearly the full range of generations. The highest generation of dominant strategy achieved for experiments 3, 4 and 5 were 176, 159 and 221 respectively. This would indicate that evolutionary progress in these experiments stagnated quite early on. These were the three experiments that introduced Cross-Validation into the evolutionary process while allowing players up to 240 moves each. The results for experiments 6 to 8, (up to 120 moves per player), were similar to the first 5 experiments, except that experiment 6 (standard co-evolution ) highest dominant strategy was generation 205.

Seed network compatibility distance:

As mentioned in the Research Methods chapter, a population of networks was created from a single 'seed' network. It was decided to store the seed network so that each new network could have its compatibility with the seed network calculated. This allowed the degree of divergence relative to the seed network to be monitored as evolution progressed. The compatibility distance was calculated every generation for the fittest network in each species, providing information on the how the process of evolution diverges, ( or not) from the original seed network. This was performed for diagnostic purposes, but produced some intriguing results. Figure 10 shows the seed compatibility distance from experiment 1 for
species that survived for 50 or more generations.

![Graph showing Compatibility Distance of Species from Seed (Experiment 1)](image)

*Figure 10: Compatibility distance from seed, experiment 1.*

Each colour represents the compatibility distance from the seed for a particular species. The various species appear and disappear as evolution proceeds. Along the x-axis of the graph, the generations at which dominant strategies appeared are marked. The general shape of the graph shows that after an initial rapid divergence from the seed, the networks settle down into a more steady divergence. The initial rapid divergence coincides with the initial burst of dominant strategies; this pattern was repeated in all of the experiments. It should also be noted that all species follow the same general line of the curve, none of them branch off to form separate branches of evolution. In experiment 4 however, this did happen, a group of species formed a separate branch at the 100 generation mark, diverging from the seed at a lower rate but became extinct after about 70 generations.

In experiment 1, something quite interesting happened. To understand it, it is necessary to know that the co-evolution software has a mechanism for restarting evolution if progress stagnates. Stagnation was judged to have occurred if, over a period of 50 generations, there is no network with a new best ever fitness score, (a count was maintained of the number of generations since a new best ever fitness score occurred). This method of judging when stagnation occurs in NEAT pre-dated the introduction of the dominance
When stagnation was detected in this way, the two fittest networks from the best two species were retained and all the other networks discarded. Evolution then proceeded in the next generation as normal, while the two surviving networks each generated half of the next generations population. This then provided for an evolutionary 'bottleneck' or aggressive selection in the hope that the evolution process would be invigorated. Figure 11 shows a detail taken from the graph shown in Figure 10.

Figure 11: Detail from 'Compatibility distance from seed, experiment 1'.

At the point $A$, stagnation occurred and just two networks survived to form the next generations' population. The steady divergence of the compatibility from the seed levelled off and stayed between the values for compatibility of 50 to 65. The species whose points are marked with triangles represents the species that would produce the next dominant strategy, which occurred as indicated at the cross-hairs. In the very next generation, at point $B$, stagnation was again deemed to have occurred, apparently the new dominant strategy didn't also produce a new best ever fitness score. Fortunately, the new dominant strategy was one of the two networks that survived to form the new population. After this point, it can be seen from Figure 10 that the population again starts to diverge from the seed network. The point to take note of is that it was the occurrence of a dominant strategy that appeared to be associated with restarting the divergent behaviour, not the mechanism.
for recognising stagnation, which occurred twice. In this particular case the new dominant strategy seems to express something that is happening within the co-evolutionary dynamics. Although by no means conclusive it may be worthwhile investigating further the relationship between dominant strategies and co-evolution.

This section has confirmed the successful occurrence of the co-evolutionary process in the experiments. The following section will examine sample games actually played during the experiments.

5.2 Trap-the-Cap games

It was not the purpose of this thesis to find optimum Trap-the-Cap players, none the less, it is appropriate to make a few general comments about the games played. It was possible to save games to a file with all of the moves recorded, however, only a very small number of games were stored as samples. Every 10\textsuperscript{th} generation, a game was selected and stored, so an experiment produced 50 sample games.

The most common strategy for networks, was to bring out one cap only until it was captured, when another cap would be brought into play to replace it. This led to most games only having two caps in play at any given time. This way of playing was most prevalent in the experiments where up to 240 moves per player were allowed and became the only way games were played after about 100 generations. In experiments where up to 120 moves per player were allowed, there were more games which held more interest, especially in the early stages of evolution. One game had up to 7 caps deployed at the same time and showed distinct tactical interplay. One player brought 4 caps into play and kept them close to the home base. The other player brought 1 or 2 caps into play at a time and moved them to be amongst the other players caps, whenever possible selecting a safe haven space to occupy.

It would appear that by varying the maximum number of moves that could be made in a game, the type of network evolved was altered. Longer games favoured networks that played safe and only deployed one cap at a time. Shorter games played in this way only reached a winning situation about 30\% of the time; networks that took more risks and deployed more caps could win quite quickly.
None of the networks discovered the ability to return to the home base to unload captured caps and hence restore own-player caps to the game. Perhaps there was no evolutionary pressure for this to happen. Possibly reducing the maximum number of games still further would prompt the discovery of this tactic as it would allow higher risk taking, since a player would have the opportunity of restoring its captured caps to the game.

This section has examined some sample games drawn from the experiments. The next section will examine if Cross-Validation made a difference to the co-evolutionary results.

### 5.3 Network Complexity

Network complexity was simply determined by counting the number of nodes and links, this was done for the dominant strategies found in the experiments. The number of nodes increased slowly over the generations from the starting complement of 147. There was no discernible difference between any of the networks; typically about 12 nodes were added to the initial starting set of nodes (+/- 2 nodes) with no pattern related to the use of Cross-Validation. So over the 500 generations the number of nodes per network was essentially the same. The number of links are shown in Figure 12 for each of the experiments. There are some differences that appear, but again there is no pattern related to the use of Cross-Validation.
Figure 12: Number of links in dominant strategy networks

5.4 Dominant Strategy Play-off's

'Dominant Strategy Play-off's ' were performed to determine if Cross-Validation gave any advantage; play-off's compete all dominant strategies from one experiment against all dominant strategies from a second experiment using the same method of competition used to find the generation champions found in section 5.1. This can give an indication of the relative performance between the experiments over the generations. The results are given in matrix form, (see Table 5 for instance), where each value represents the winning margin for each dominant strategy combination. A score can be positive or negative. A positive value is recorded if the experiment represented by the rows won and a negative value if the experiment represented by the columns won. The shaded cells divide each matrix up into two halves; the shaded cells should see wins for the column's dominant strategies, while the unshaded cells should see wins for the row's dominant strategies. This assumes that networks from later generations will defeat networks from earlier generations. So, if the play-off goes according to generation, the shaded cells will contain negative values and the unshaded cells positive values. Where a result has gone against expectations, the value in the cell is highlighted in red. In the matrices presented, rows always represent experiments that included Cross-Validation, while columns represent standard co-evolution experiments.
Experiments with 240 moves per player per game before draw is declared:

Experiments 3, 4 and 5 all used Cross-validation during evolution, but in varying amounts. These experiments were each put into a play-off with experiment 2, which evolved without the use of Cross-Validation. Experiment 1 was not used in the play-off's since it was the source of networks for the Cross-Validation of experiments 3, 4 and 5. The play-off results are presented in Table 5, Table 6 and Table 7.

The first play-off, Table 5, pitted the experiment that obtained 25% of its CM networks from a Cross-Validation CM against a standard co-evolution experiment. The only part of this table that can be said to conform to expectations is the first column, where all experiment 3 networks defeat the 1st generation network of experiment 2. The rest of the table fails to show either experiment gaining an advantage or indeed either experiment winning all of the games it should win just by considering relative generation. In particular, the later generations of both experiments are defeated by much earlier generations. This would suggest that, although the strategies learnt by each experiment have improved over the course of evolution, as evidenced by the appearance of dominant strategies, neither set of strategies is particularly good in a global sense.

<table>
<thead>
<tr>
<th>Experiment 2 (co-evolution)</th>
<th>1</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>8</th>
<th>16</th>
<th>27</th>
<th>39</th>
<th>62</th>
<th>65</th>
<th>376</th>
<th>453</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dominant Strategies</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>-4</td>
<td>0</td>
<td>-9</td>
<td>-6</td>
<td>-12</td>
<td>6</td>
<td>-10</td>
<td>-8</td>
<td>-23</td>
<td>-16</td>
<td>-2</td>
</tr>
<tr>
<td>3</td>
<td>23</td>
<td>-6</td>
<td>13</td>
<td>0</td>
<td>-9</td>
<td>6</td>
<td>-2</td>
<td>-4</td>
<td>8</td>
<td>13</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>17</td>
<td>1</td>
<td>2</td>
<td>-6</td>
<td>-6</td>
<td>-1</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>-3</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>17</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>-1</td>
<td>9</td>
<td>-7</td>
<td>-1</td>
<td>-5</td>
<td>-4</td>
<td>-3</td>
</tr>
<tr>
<td>12</td>
<td>19</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>5</td>
<td>7</td>
<td>2</td>
<td>8</td>
<td>-2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>9</td>
<td>3</td>
<td>-3</td>
<td>10</td>
<td>1</td>
<td>7</td>
<td>2</td>
<td>-6</td>
<td>-6</td>
<td>-4</td>
<td>-6</td>
</tr>
<tr>
<td>31</td>
<td>17</td>
<td>0</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>0</td>
<td>3</td>
<td>5</td>
<td>-3</td>
</tr>
<tr>
<td>49</td>
<td>13</td>
<td>11</td>
<td>14</td>
<td>-5</td>
<td>-1</td>
<td>-3</td>
<td>-4</td>
<td>6</td>
<td>-2</td>
<td>-5</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>93</td>
<td>15</td>
<td>-1</td>
<td>6</td>
<td>-2</td>
<td>4</td>
<td>-1</td>
<td>-4</td>
<td>1</td>
<td>-4</td>
<td>2</td>
<td>-3</td>
<td></td>
</tr>
<tr>
<td>165</td>
<td>19</td>
<td>-3</td>
<td>7</td>
<td>1</td>
<td>-10</td>
<td>0</td>
<td>4</td>
<td>-5</td>
<td>6</td>
<td>-7</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>176</td>
<td>15</td>
<td>-3</td>
<td>6</td>
<td>-1</td>
<td>-8</td>
<td>-3</td>
<td>5</td>
<td>7</td>
<td>5</td>
<td>1</td>
<td>-2</td>
<td>-1</td>
</tr>
</tbody>
</table>

Table 5: Dominant strategy play-off scores (Expt 3 vs Expt 2)
The second play-off, Table 6, pitted the experiment that obtained 50% of its CM networks from a Cross-Validation CM against a standard co-evolution experiment. This result looks much more as if Cross-Validation has had a beneficial effect on the evolutionary process. Consider experiment 4 (rows) and ignore for the time being rows 100 and 159; only 3 competitions go against the run of play in the unshaded cells and two of those are where the relative generations taking part are quite close (generation 5 vs 4 and 9 vs 8). However, the shaded cells have 20 competitions going against the run of play and they occur where there are significant differences in competing generations, (generation 27 defeating generation39, 62 and 65 for example). When rows 100 and 159 are considered it appears that the situation has deteriorated and the Cross-Validation experiment has lost its advantage.

<table>
<thead>
<tr>
<th>Experiment 2 (co-evolution)</th>
<th>Experiment 4 (Cross-Validation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dominant Strategies</td>
<td></td>
</tr>
<tr>
<td>1  2  -4  3  -3  -4  -10  -9  -13  -2  -10  -8</td>
<td>1  23  5  8  6  -3  4  0  -4  -7  3  6  9</td>
</tr>
<tr>
<td>2  13  8  6  1  1  -4  3  -7  -5  -1  -12  -2</td>
<td>16  10  2  10  4  -2  6  1  -4  1  -1</td>
</tr>
<tr>
<td>5  15  -8</td>
<td>0  -3  -4  -14  -2  -7  -6  0  0  -2</td>
</tr>
<tr>
<td>9  23  5  8  6  -3  4  0  -4  -7  3  6  9</td>
<td>27  10  11  4  3  7  1  -3  1  6  0  1  -10</td>
</tr>
<tr>
<td>16  10  2  10  4  -2  6  1  -4  1  -1</td>
<td>29  15  7  4  2  -3  -3  0  -5  -3  0  3  -1</td>
</tr>
<tr>
<td>100 20  12  3  3  13  5  10  -8  4  -4  -6  -5</td>
<td>100 20  12  3  3  13  5  10  -8  4  -4  -6  -5</td>
</tr>
<tr>
<td>159 22  7  5  4  -5  3  7  -10  -4  -2  -1  -8</td>
<td>159 22  7  5  4  -5  3  7  -10  -4  -2  -1  -8</td>
</tr>
</tbody>
</table>

*Table 6: Dominant strategy play-off scores (Expt 4 vs Expt 2)*

The third play-off, Table 7, pitted the experiment that obtained 75% of its CM networks from a Cross-Validation CM against a standard co-evolution experiment. The first column shows all of the experiment 5 strategies defeating the first network of experiment 2 as expected. The first two rows go according to expectations. The next 5 rows, (generations 3, 4, 5, 9 and 22), show the Cross-Validation experiment exerting a slight advantage over the co-evolution experiment apart from one competition between generation 22 and 5, although some of this advantage can perhaps be attributed to a weakness in the later generations of experiment 2 (generations 376 and 453). From the dominant strategy in row 33 onwards there cannot said to be any advantage gained by either experiment, with a
definite weakness appearing in the last two generations, (162 and 221) as for experiment 2.

### Experiment 2 (co-evolution)

<table>
<thead>
<tr>
<th>Dominant Strategies</th>
<th>1</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>8</th>
<th>16</th>
<th>27</th>
<th>39</th>
<th>62</th>
<th>65</th>
<th>376</th>
<th>453</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>-9</td>
<td>-16</td>
<td>-15</td>
<td>-13</td>
<td>-18</td>
<td>-9</td>
<td>-10</td>
<td>-20</td>
<td>-11</td>
<td>-8</td>
<td>-4</td>
</tr>
<tr>
<td>3</td>
<td>19</td>
<td>4</td>
<td>-5</td>
<td>-10</td>
<td>-13</td>
<td>-4</td>
<td>-7</td>
<td>-1</td>
<td>-2</td>
<td>-13</td>
<td>-3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>-1</td>
<td>1</td>
<td>-5</td>
<td>-6</td>
<td>-9</td>
<td>-8</td>
<td>-11</td>
<td>-3</td>
<td>-5</td>
<td>3</td>
<td>-4</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>4</td>
<td>-2</td>
<td>-16</td>
<td>-8</td>
<td>-10</td>
<td>-8</td>
<td>-10</td>
<td>-7</td>
<td>3</td>
<td>-6</td>
<td>-6</td>
</tr>
<tr>
<td>9</td>
<td>15</td>
<td>7</td>
<td>0</td>
<td>2</td>
<td>-2</td>
<td>-3</td>
<td>3</td>
<td>-1</td>
<td>-3</td>
<td>4</td>
<td>4</td>
<td>-8</td>
</tr>
<tr>
<td>22</td>
<td>14</td>
<td>7</td>
<td>-1</td>
<td>2</td>
<td>0</td>
<td>8</td>
<td>4</td>
<td>1</td>
<td>-1</td>
<td>-8</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>33</td>
<td>19</td>
<td>11</td>
<td>3</td>
<td>-9</td>
<td>-5</td>
<td>0</td>
<td>-4</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>50</td>
<td>13</td>
<td>1</td>
<td>-2</td>
<td>4</td>
<td>1</td>
<td>-5</td>
<td>-7</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>-1</td>
<td>3</td>
</tr>
<tr>
<td>54</td>
<td>12</td>
<td>1</td>
<td>4</td>
<td>-8</td>
<td>6</td>
<td>-7</td>
<td>-2</td>
<td>3</td>
<td>6</td>
<td>-4</td>
<td>1</td>
<td>-10</td>
</tr>
<tr>
<td>87</td>
<td>12</td>
<td>0</td>
<td>-5</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>6</td>
<td>0</td>
<td>8</td>
<td>4</td>
<td>1</td>
<td>-4</td>
</tr>
<tr>
<td>98</td>
<td>17</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>-7</td>
<td>3</td>
<td>5</td>
<td>-4</td>
<td>8</td>
<td>-2</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>162</td>
<td>19</td>
<td>5</td>
<td>-4</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>-1</td>
<td>-9</td>
<td>-1</td>
<td>5</td>
<td>-1</td>
<td>4</td>
</tr>
<tr>
<td>221</td>
<td>19</td>
<td>11</td>
<td>7</td>
<td>7</td>
<td>-1</td>
<td>1</td>
<td>7</td>
<td>2</td>
<td>-9</td>
<td>8</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

*Table 7: Dominant strategy play-off scores (Expt 5 vs Expt 2)*

### Experiments with 120 moves per player per game before draw is declared:

Experiment 8 used Cross-validation during evolution, with 50% of its CM networks drawn from a Cross-Validation CM. Experiment 7 was a standard co-evolution experiment.

The play-off, Table 8, pitted experiment 8 against experiment 7 and produced the clearest results of all of the experiments. Generations 1, 4, 6 and 51 all demonstrate clear superiority over experiment 7, with generation 19 performing as expected. As with the previous results, the last 3 dominant strategies of experiment 8 don't perform as well as the earlier dominant strategies, but their performance is still close to that expected.

The next section will present the Master Tournament results and the subsequent best-of-run networks, along with best-of-run play-off's to establish which experiments are 'best'.
### Table 8: Dominant strategy play-off scores (Expt 8 vs Expt 6)

<table>
<thead>
<tr>
<th>Experiment 7 (co-evolution)</th>
<th>Experiment 8 (Cross-Validation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dominant Strategies</td>
<td>1</td>
</tr>
<tr>
<td>----------------------------</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>45</td>
</tr>
<tr>
<td>4</td>
<td>46</td>
</tr>
<tr>
<td>6</td>
<td>41</td>
</tr>
<tr>
<td>19</td>
<td>46</td>
</tr>
<tr>
<td>51</td>
<td>44</td>
</tr>
<tr>
<td>141</td>
<td>40</td>
</tr>
<tr>
<td>218</td>
<td>42</td>
</tr>
<tr>
<td>292</td>
<td>39</td>
</tr>
</tbody>
</table>

### 5.5 Best of Run Play-offs

A Master Tournament was performed for each evolution experiment to find the Best of Run network, i.e. the best network from the evolutionary process. A Master Tournament plays every generation champion against every other generation champion and declares the network that wins the most games as the winner. In the Dominance Tournament, each competition between two dominant strategies consisted of a total of 200 games, as outlined in section 5.1 for finding generation champions. If this number of games were played per competition in the Master Tournament then it would mean a total of 24 million individual games would have to be played; each Master Tournament would take about 14 days to run. To bring the execution time down to a more reasonable level, just 8 games were played per competition between generation champions. This gave a total of one million individual games, or about 14 hours execution time.

The results of the Master Tournament are shown in Table 9 where the generation champion that won each Master Tournament is shown against its experiment number. Also shown for comparison is the generation of the nearest dominant strategy. Experiment 1 and 5 have a very close match between Master Tournament winner and a dominant strategy; experiments 7 and 8 are within 23 and 24 generations respectively; experiments 2, 3, 4 and 6 are all more than one hundred generations from the nearest dominant strategy. It can be
seen that there is not necessarily a close relationship between Master Tournament winner and a dominant strategy.

<table>
<thead>
<tr>
<th>Experiment Number</th>
<th>Master Tournament Winner (generation)</th>
<th>Nearest Dominant Strategy (generation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>390</td>
<td>393</td>
</tr>
<tr>
<td>2</td>
<td>230</td>
<td>376</td>
</tr>
<tr>
<td>3</td>
<td>307</td>
<td>176</td>
</tr>
<tr>
<td>4</td>
<td>447</td>
<td>159</td>
</tr>
<tr>
<td>5</td>
<td>93</td>
<td>98</td>
</tr>
<tr>
<td>6</td>
<td>314</td>
<td>205</td>
</tr>
<tr>
<td>7</td>
<td>337</td>
<td>315</td>
</tr>
<tr>
<td>8</td>
<td>316</td>
<td>292</td>
</tr>
</tbody>
</table>

*Table 9: Master Tournament Winners*

Once the best-of-run networks have been found, they can compete to find which is the better. Each of the networks that evolved using Cross Validation were competed with the two standard co-evolution networks from the same series of experiments. The competition was run in the same way as described in section 5.1, except that 2000 games were played instead of 200. This was possible since there were only 8 competitions. Again, the values shown in the results table represent the winning margin of the competition. A positive value is recorded if the experiment represented by the rows won and a negative value if the experiment represented by the columns won. Table 10 and Table 11 show the results for the experiments with 240 and 120 moves per player per game before draw is declared respectively.

<table>
<thead>
<tr>
<th>Cross Validation</th>
<th>Experiment</th>
<th>Co-evolution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>35</td>
<td>-11</td>
</tr>
<tr>
<td>4</td>
<td>52</td>
<td>37</td>
</tr>
<tr>
<td>5</td>
<td>-10</td>
<td>-36</td>
</tr>
</tbody>
</table>

*Table 10: Best-of-run play-off,( 240 moves per player)*
It can be seen that the experiments which obtain 50% of their CM networks from a Cross-
Validation CM, (experiments 4 and 8), always win the best-of-run competition with the 
standard co-evolution experiments.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
</table>
| Cross-
Validation | 8 | 19 | 45 |

Table 11: Best-of-run play-off,( 120 moves per player)
6 Discussion and Conclusions

This thesis aims to answer whether the use of Cross-Validation during the co-evolution of neural networks provides measurable benefits in terms of speed of evolution, network complexity and performance when compared to standard co-evolution.

Starting with the question of performance: The best indications that there was a performance benefit were given by experiments (or populations) 4 and 8. Firstly, the best-of-run networks for these two populations each defeated their two respective best-of-run networks that were evolved under similar conditions but without the inclusion of Cross-Validation, (populations 1 & 2 and populations 6 & 7). So populations 4 and 8 were superior to standard co-evolution networks. The picture was more complex when the dominant strategy play-offs were considered. They showed that the main advantage populations 4 and 8 enjoyed, occurred early on in the co-evolutionary process, below about 100 to 150 generations. After that, they were more evenly matched with the standard co-evolution populations. So there was an early advantage, but it seemed to diminish as evolution progressed. The maximum number of moves allowed in the Trap-the-Cap game before a draw was declared appeared to be an important factor. When this value was reduced from 240 moves for population 2, to 120 moves for population 4, it was possible to see that population 4 was performing better in the later generations compared to population 2. It was also noted that the type of game play evolved by the networks under the two different move limits changed; more risks were taken by the networks in population 4 in order to finish games more quickly, although interestingly, this behaviour almost disappeared in later generations. So a performance benefit was observed in the early generations, but it is hypothesised that in the later generations, a very safe mode of play won out, rendering the Cross-Validation and co-evolution populations as equals.

Next consider the speed of evolution: The progress of evolution is measured by the appearance of dominant strategies in a population. The evolution of Cross-Validation populations 3, 4 and 5 achieved no dominant strategies higher than 221; this was in contrast to the standard co-evolution populations 1 and 2 which achieved a dominant strategy at generation 393 and 453 respectively. The Cross-Validation population 8 did slightly better with a dominant strategy at generation 292. So the Cross-Validation populations stagnated early. However, the dominant strategy play-off matrices for...
populations 4 & 2, and populations 8 & 6 show that (below ~100 generations) the Cross-Validation networks (4 & 8) could quite often, defeat standard co-evolution networks (2 & 6) from later generations. So the speed of evolution was greater for the Cross-Validation experiments, at least until evolution stagnated. One point that it is worth considering is that the game of Trap-the-Cap may not be an interesting game given too many moves and optimal conservative play is achieved quickly giving different approaches nothing to differentiate themselves.

Finally, consider the question of network complexity: The complexity of the network was judged on the number of links, which turned out to show no particular correlation between populations using Cross-Validation and those that were not. The same argument used for speed of evolution can be applied to complexity. When Cross-validation networks defeat standard co-evolution networks from later generations, then the former have fewer links and are therefore achieving more for less complexity.

There was a second research question: What proportion of Cross-Validation CM members should the co-evolutionary process use? This was addressed by using different proportions of Cross-Validation networks in the set of 'testers' during co-evolution, 25%, 50% and 75%. It is difficult to say that a positive or negative result was achieved since the data was inconclusive and there was not enough time to run more experiments to clarify the issue. In the one population that used 25% Cross-Validation and the one population that used 75%, the results indicated that they both did not perform as well as the populations that used 50%. So perhaps 50% is the best proportion, but more experiments need to be completed.

Further work should obviously concentrate on performing more co-evolutionary experiments to obtain statistical bounds on the answers to the questions, something that was not valid for this thesis given the low number of experiments that could be performed in the time. Also, different problems should be tried to see if the same patterns of evolutionary stagnation occur or if they are related to the game of Trap-the-cap.

There were a number of interesting points that could be investigated further. Firstly, the graph showing species compatibility distance from the seed network, with one exception, showed all species following just one curve. So even with speciation, they were all
following a similar path when perhaps other evolutionary paths could be available. The
one exception showed a separate branch developing, until it became extinct after about 70
generations. It would be interesting to investigate conditions that produce and encourage
branching. Secondly, it was observed that the appearance of dominant strategies might be
an expression of the underlying co-evolutionary process and it would be interesting to
investigate any correlation between dominant strategies and the dynamics of co-evolution.
References


53


## Index

| A | Actuators................................. 2 |
|   | Ancestral members...................... 21 |
|   | Arms race................................ 6, 13 |
| B | Best of Generation...................... 22 |
|   | Best of Run............................. 22, 24 |
| C | CM........................................... 6, 16 |
|   | Co-evolutionary Memory............... 6, 13, 16, 24 |
|   | Compatibility........................... 27 |
|   | Compatibility distance............... 38 |
|   | Compatibility threshold............... 15 |
|   | Competitive Co-evolution............. 6, 13, 19 |
|   | Competitive fitness sharing.......... 13, 20 |
|   | Complexification....................... 16 |
|   | Connection................................ 4 |
|   | Cross-Validation....................... 7, 24, 29 |
|   | Crossover................................ 4, 8, 21 |
|   | Crowding................................ 14 |
|   | Cyclic rediscovery..................... 8 |
| D | Dominance Tournament................. 9, 21, 24, 30 |
|   | Dominant strategy...................... 21 |
|   | Dominant Strategy Play-off........... 43 |
| E | Enhanced fitness score............... 15 |
|   | Equal Effort................................ 22 |
|   | Evolutionary pressure................ 19 |
|   | Evolutionary programming............... 11 |
| F | Experiments.................. 30 |

| Fitness........................................... 4 |
| Fitness function............................ 19 |
| Fitness landscape.......................... 14 |
| Fitness sharing.............................. 6, 13, 14 |
| FS-NEAT......................................... 13 |
| Generation.................................... 26 |
| Generation champion....................... 16 |
| Generation number......................... 13 |
| Genetic algorithm.......................... 3 |
| Genetic algorithms.......................... 3 |
| Global progress.............................. 7 |
| Hall of fame................................... 9, 16, 20 |
| Hard-limited.................................. 32 |
| Historical progress......................... 6 |
| Input signals................................. 4 |
| Input values................................... 28 |
| Interconnection weights.................... 4 |
| Java............................................ 24 |
| Local progress............................... 6 |
| Master Tournament......................... 9, 22, 30 |
| Mutation........................................ 4, 21 |
| NEAT............................................ 8, 12, 20 |
| Network Complexity.......................... 21, 23 |
| Network Performance....................... 21, 23 |
| Neural network.............................. 4 |
Neuro Evolution of Augmenting
Topologies........................................12
Node..................................................4
Node and Link count............................23
Optimal behaviour..............................14
Output connections............................4
Output values....................................28
Parameter values................................30
Progress...........................................17
Rational agent....................................2
Recurrent neural networks....................11
Red Queen effect.................................21
Reinforcement learning.........................3, 10, 19
Reproduction.....................................26
Seed network....................................38

Sensors..............................................2
Shared sampling.................................13, 20
Software...........................................24, 30
Speciation...........................................6, 13, 15
Speed of Evolution............................21, 23
Stagnation..........................................39
State...................................................28
Strategy.............................................5
Superiority.........................................17
Supervised learning............................2
TDGammon.........................................10
Testers...............................................32
Trap-the-cap.......................................1, 3
Unsupervised learning........................2
Weights..............................................4
Appendix A: Extended Abstract
Cross-Validation of Fitness Scores During Co-evolution
Using the 'Trap-the-Cap' Board Game as a Testbed
C J Flynn

Extended Abstract of Open University MSc Dissertation Submitted 9 March 2010

Introduction
Games have always been used as a convenient way of testing techniques from Artificial Intelligence (AI), they have well defined rules and well defined outcomes and can be used as a reference between research groups. The field of AI covers a wide range of methods and techniques, all with the aim of producing systems, that to some extent or other, are able to perceive, reason and act. The concept of an 'agent' acting in the problem space encapsulates all AI solutions as having, in some form, the ability to perceive an environment using 'sensors' and some means of influencing the environment using 'actuators'. The various sub-fields of AI provide the bit in between the sensors and the actuators; the reasoning, and possibly some learning as well.

A board game called Trap-the-Cap is used to investigate the evolution of a software agent that can learn to play the game without any prior knowledge and without the help of a teacher. The only stimulus used to guide the agent is the score of how many games it wins. This comes under the general heading of reinforcement learning. The reinforcement is supplied by co-evolving two populations of agents in tandem, each taking it in turns to test the other. The populations start out as completely naïve Trap-the-Cap players and gradually increase in sophistication over the ensuing generations. This is an important area that can help to move AI away from bespoke solutions towards applications that can be applied to different problems with less specialist AI knowledge.

Generally when learning it is necessary to have independent training and testing functions. This is difficult for co-evolution since each population is the trainer and tester for the other population. This thesis investigates the technique of injecting independent test agents into the co-evolution cycle to provide 'Cross-Validation' of the testing of a population. It asks the question 'does the use of Cross-Validation provide measurable benefits in terms of speed of evolution, network complexity and performance?'

Results
Neural networks were used as the agents in the two populations. The particular technique used to evolve and mutate them is called Neuro-Evolution of Augmenting Topologies (NEAT) which is a method that represents neural networks as a genetic code and allows the use of the standard genetic operations of cross-over and mutation. Before the introduction of NEAT, authors generally thought that neural networks could only be mutated. Cross-Validation was achieved by providing a source of independently evolved neural networks as an additional source of testers during the co-evolution cycle.

It was important to establish that co-evolution was occurring, this was done using the concept of Dominant Strategies. A dominant strategy is a neural network that is the
champion of its generation and can also beat all earlier dominant strategies; the first dominant strategy is defined as the generation champion of the first generation. The emergence of dominant strategies define where co-evolution is continuing or if the populations have stagnated. The following graph shows that co-evolution did indeed take place over the 8 experiments performed.

The experiments shown in the above graph are marked as being 'CM' which means standard co-evolution only or as CM+CV which means co-evolution with the addition of Cross-Validation. The results indicate that the CM+CV experiments all stagnated, i.e. stopped any significant evolution, much earlier than the CM experiments.

The merits of using Cross-Validation were tested by competitions between experiments that used cross-Validation and those that did not. The competitions were conducted at different generations, so one generation from one experiment would be competed against the full range of generations from another experiment. In this way matrices of results were built up that gave a historic breakdown of the relative strength of each experiment as it evolved. The results all showed the same pattern: Experiments using Cross-Validation established an early superiority over the standard co-evolution experiments, but which disappeared after about 100 generations.

Analysis

Results were encouraging and showed that there was indeed an advantage gained by Cross-Validation in terms of performance, speed of evolution and network complexity. The disappearance of this advantage after the first hundred or so generations was possibly explained by observing the games played by the neural networks. In earlier generations there were quite a few risk taking networks that could win quite quickly, but as evolution progressed, these were replaced by very conservative players who took few risks even at the expense of having games declared a draw, due to hitting the maximum allowed move limit. It didn't matter which method was used to evolve a network, in the end they all
ended up using the same strategy.

**Discussion**

It would seem that the introduction of Cross-Validation into the co-evolution process does improve evolved agents at a given generation when compared to standard co-evolution. However, in these experiments, the improvement in the Cross-Validated agents allowed them to stagnate at a particular level of play quite quickly; a level of play that the standard co-evolved agents could catch up given more generations.

Improvements to AI methods will always have importance across many application domains that are too complex to rely on pre-programmed solutions. Co-evolution is an important technique when there is not enough knowledge to provide an agent with the required training. However, co-evolution is subject to certain 'pathologies' such as rediscovering successful strategies from earlier generations and then simply cycling through them, or only exploring a small part of the problems state space. Being able to inject independently gained experience into the co-evolution process could help to make it more reliable and stable as a technique.