Analysing and Exploiting Transitivity to Coevolve Neural Network Backgammon Players

Mete Çakman
Dissertation for Master of Science in Artificial Intelligence and Gaming
Universiteit van Amsterdam
August 1, 2008

Abstract

This thesis investigates using coevolution for training neural networks to play the game of backgammon. We analyse the usefulness of coevolution in this domain, compare results of "round robin", "fitness sharing", and "hall of fame" coevolution techniques, and make a thorough analysis of the transitivity and rank distribution of individuals in a single evolving population. We find that the backgammon domain is highly transitive, and that 50% of the time during coevolution a newly evolved individual will be the worst member of the population, with the other 50% evenly distributed over all other population ranks. We attempt to exploit this analysis through three new fitness evaluation schemes. "Binary rank placement" uses a binary search to calculate individuals' ranks, "single evaluator" uses a single individual taken from the evolving population to evaluate fitness levels, and "losers first" assesses individuals against the worst in the population first, aborting evaluation if the match is lost in order to prevent wasting fitness tests. We find that only the losers first scheme provides an increase in efficiency. Finally, we use the losers first method to try to evolve more sophisticated nonlinear network structures, in an attempt to outperform previous work using coevolution for optimisation in the backgammon domain. We discover that the domain can be exploited for more efficient fitness evaluation, yet are unable to evolve superior nonlinear solutions with the current experimental setup.

Acknowledgements

Many thanks to my supervisor, Shimon Whiteson, whose energy and attention kept me motivated and interested in my work, week after week, and whose observations and suggestions were vital to the direction of this thesis.

Many thanks also to Gerry Tesauro for his assistance and invaluable C code.

Thanks to Rogier Koppejan for his easy to read, use, and maintain NEAT implementation in C++.

Thanks to my study-partner-in-crime Corrado Grappiolo for sharing the same boat, and providing hours of conversational distractions.

Finally, all of this was made possible by the wonderful people at NUFFIC, who provided me with a scholarship to study here in Amsterdam.

All experiments in this thesis were run on the computer cluster facilities kindly provided by the SARA Computing and Networking Services here in the Netherlands.

Title page image and backgammon layout in Chapter 3 taken from Wikipedia.

Contents

List of Figures

1 Introduction

Backgammon is a game for two players involving skill and luck that has been a focus for studies in artificial intelligence (AI) since the late 1970's. Computer programs have been taught to play using human knowledge databases, hill-climbing optimisation algorithms, evolutionary computation, and reinforcement learning techniques, yet our understanding of why some techniques work better than others remains incomplete. In this thesis, we investigate using evolutionary computation methods for optimisation in the backgammon domain, comparing and analysing different strategies and exploiting our results to develop more efficient methods of evolution of backgammon players.

Previous research efforts using AI in the backgammon domain include Tesauro's TD-Gammon program [18], Pollack & Blair's hill-climbing optimisation algorithm [10], and Darwen's work in evolutionary computation [2], all of which involved training neural networks to evaluate backgammon play. Tesauro used temporal difference (TD) learning, a form of learning which predicts future returns in order to update current value estimations, to create a formidable backgammon player that learnt to play at a master level, surpassing previous backgammon programs and displaying strategies that have improved on expert human play [18]. Pollack & Blair achieved surprising results using a naïve hill-climbing optimisation algorithm which, despite playing a good intermediate level game, suffers from a low plateau in skill level. Darwen used coevolution to train neural networks, by evolving players whose fitness evaluations are based on competition with other networks of the same evolving population, and compared his results with those of Tesauro. Darwen achieved a high standard of play, surpassing TD-learning for simple linear network structures, yet failed to evolve any nonlinear structure necessary for more advanced play, apparently due to infeasible computation times [2]. Darwen did not look at methods for more efficient coevolution to try to surpass these limitations, however the work done by Pollack & Blair suggests that the backgammon domain is highly conducive to coevolutionary strategies, and Tesauro demonstrates that neural networks are capable of playing backgammon at a master level.

This thesis investigates coevolution in the backgammon domain, analysing the domain and attempting to more efficiently evolve game strategies in order to surpass the limitations observed by Darwen. The thesis consists of three main parts – an initial investigation into the usefulness of coevolution for training backgammon players and a comparison of coevolutionary strategies, followed by an analysis of domain transitivity (a domain is *intransitive* if cycles of expertise exist such that agent A beats agent B, agent B beats agent C, but agent C beats agent A), and finally the implementation and analysis of new fitness evaluation strategies for more efficient coevolution.

In our first experimental chapter we investigate the usefulness of evolution for optimising backgammon players, comparing different population sizes to compare true evolution with Pollack & Blair's hill-climbing optimisation, a pared down form of evolution with a population size of just 2. We examine the benefits of coevolution, where individuals are evaluated on an evolving set of tests, over evolution, which uses a fixed fitness evaluation. We then compare coevolutionary strategies using *round robin* tournaments, as used by Darwen, to *fitness sharing* and *hall of fame* techniques, designed to maintain diverse teaching sets for better evolution in the presence of intransitivities.

The results of the more advanced strategies of fitness sharing and hall of fame show no improvement over the round robin approach. Because these methods are designed for better coevolution in intransitive domains, our next chapter investigates whether the domain is in fact transitive, despite intransitivities found in the backgammon domain by Pollack & Blair. We discover that these intransitivities do not exist in the true domain, but are caused by noise in the fitness evaluation used by Pollack & Blair, which explains why no improvement was gained through fitness sharing and hall of fame techniques.

The final experimental chapters use this knowledge to investigate new strategies for reducing the number of evaluations required for coevolution in the backgammon domain. "Binary rank placement" uses a binary search to find correct rankings within a population, and we discover that this fares worse than round robin due to the inherent noise in the backgammon fitness evaluation. "Single evaluator" uses a single backgammon player from within the population to test fitness values. However, inferior teacher selection capability causes it to be less efficient as well. Finally, we analyse the distribution of fitness rankings for the round robin strategy and discover that new individuals are unhelpful to evolution 50% of the time. We exploit this with the "losers first" strategy by testing against the worst ranked player first, halting evaluation if the match is lost. This increases optimisation speed for coevolution of backgammon players. Our final experiments use the losers first strategy for coevolution of more complex nonlinear neural networks. However, computational limitations prevent us from achieving better results with these networks.

This thesis is structured as follows. Chapter 2 gives background on the AI tools used in this paper – neural networks and evolutionary computation, describing steady-state evolution and coevolution. Chapter 3 introduces the game of backgammon and discusses previous work using AI in the backgammon domain. Chapter 4 describes the initial experiments performed – population size tests, coevolution versus fixed evaluation, and a comparison of different coevolutionary strategies. A deeper analysis of domain transitivity is presented in Chapter 5, and in Chapter 6 our experiments in more efficient evaluation are described with results and analysis. Chapter 7 presents final experiments using our losers first method for both linear and nonlinear network structures. Chapter 8 concludes with a discussion of results and a final comparison to related work, as well as directions for future work.

2 Background

This chapter describes the fundamentals of the tools used in this paper: neural networks and evolutionary computation, describing the steady-state approach as well as coevolution principles.

2.1 Neural Networks

Neural networks are perhaps the oldest surviving tools of the field of artificial intelligence, dating back to the 1940's when "cybernetics," as it was called then, became a hot topic of mathematical research. Studies showed that the human brain resembled a network of electrical "neurons" which fired electrical pulses in a digital fashion, and could be modelled with electrical circuits. As computers developed in the 1950's it became possible to implement these simple models of the brain, and in 1951 Marvin Minsky and Dean Edmonds created the first neural net computer, the SNARC. Neural networks suffered a complete loss of attention in the 70's due to proofs showing that even simple functions such as XOR could not be approximated with single layer networks, and that finding optimal weights for multi-layer networks is NP-hard[1]. However, in the 1980's they began to make a comeback with work in fields other than computer science, namely physics and psychology, and the use of the backpropagation algorithm for training multi-layer networks [12]. This led to their first successful commercial applications in the 90's, in tasks such as handwriting and speech recognition [7, 12].

Neural networks are linked networks of artificial neurons, attempting to model the behaviour of the human brain. An artificial neuron is a node in such a network which accepts multiple input values, and has a single output. A neuron will first multiply each input value by a corresponding input *weight* value, sum the results, and finally pass this sum through some threshold function providing the final output value of the neuron, as demonstrated in Figure 2.1. For example, a basic threshold function might sets its output to 1 or -1 depending on whether the input value exceeds some threshold limit.

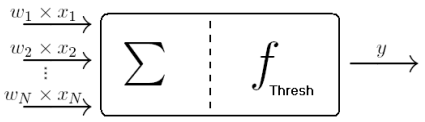


Figure 2.1: Artificial neuron, with N inputs, and a general thresholding function.

A neuron thus outputs a function of its inputs, determined by its thresholding function and its input weights. It is the weight values attributed to each input which gives a neuron, and thus a network of neurons, its ability to adapt, using for example learning algorithms which modify those weights.

[1]Ironically demonstrated by Minsky himself [6].

A neural network may be acyclic or recurrent depending on the application. One of the most widely used network structures is a fully connected feed-forward network consisting of a layer of input neural nodes, one or more layers of hidden nodes[2], and a final output node layer (see Figure 2.2.)

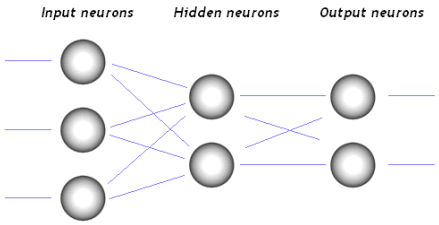


Figure 2.2: Feed-forward network of artificial neurons, all connections go from left to right.

Such a feed-forward network is able to represent any mathematical nonlinear function to arbitrary accuracy [7], meaning that theoretically any mathematical solution can be represented using a neural network. Of course, in practice this is not always the case, as the more complex the problem, the more complex the network necessary and the more weights which require optimisation, creating higher dimensional optimisation problems. However, neural networks are able to represent complex spaces of highly nonlinear functions, and as such are useful when learning functions of an unknown form a priori [7]. For this reason they are used in this paper as move evaluator functions for playing backgammon (Section 3.5 describes the use of neural networks as move evaluators in more detail).

2.2 Evolutionary Computation

For training neural networks to play backgammon this thesis uses evolutionary computation (EC). EC is a form of optimisation loosely based on the theory of evolution. The basic premise of evolutionary theory is that individual organisms live in populations and have a basic genetic code (genotype) that dictates how their actual appearance and functionality will be represented within their environment (phenotype). In each new generation, parents pass on combinations of their genetic material in varying ratios to their offspring and occasionally genetic mutations will occur. Between each generation, individuals must survive to maturity before they are able to pass on their genes. This way stronger individuals with successful genes will pass theirs on, while weaker individuals will not - thus ensuring the population keeps strong and beneficial genes in the gene-pool, and quickly sheds its unhelpful ones, a notion Charles Darwin termed *"survival of the fittest."*

When couched in these general terms it is easy to see how this process could be modelled as an adaptive algorithm to learn to solve a given problem, provided the potential solutions be expressed in terms of genotypes *and* phenotypes, that the corresponding functions of genetic crossover (breeding between 2 or more individuals) and

[2]So named as they are not visible to the user from a black-box perspective.

mutation are adequately defined, and that their fitness level can be evaluated against the problem at hand.

Because of the randomised mechanisms of mutations and parent selection, EC provides a method for searching a multi-dimensional solution-space which, though often expensive in terms of time and/or computational effort, is much less likely to get permanently stuck in local maxima than deterministic search methods [7].

As a relevant example, EC can be used to train neural networks. In the case of pure weight evolution, as used in this thesis, we start with a population of networks with identical structures and randomly generated initial weight values, encode those weight values to a genotype code, and proceed to appraise each network in light of the problem to be solved. Those that perform better are given a higher fitness value, and thus have a higher chance of being allowed to mate. Once all networks have been evaluated, some percentage of the population is discarded, while the remainder are used to create new offspring by genetic recombination and occasional mutation of those weight genotypes. This process of evaluation is repeated until a satisfactory solution is found.

2.3 Steady-State Evolution

The more traditional genetic algorithm is based on generations of individuals – evaluating an entire generation of individuals at a time, then evolving an entirely new generation and repeating. Thus, with a population size of 200, 200 evaluations would be performed, then a whole new population of 200 individuals selected and bred from the best performing individuals. However, in this thesis *steady-state* evolution is used, in which each evolutionary step consists of removing just one, typically worst, individual from the population, and breeding one from the remaining population, thereafter evaluating the new individual and moving on to the next step. This allows the same process of evolutionary computation to be carried out in smaller increments, making it possible to gauge progress on an individual scale.

2.4 Coevolution

Most experiments in this thesis are based on coevolution, whereby fitness is evaluated using other members of the same population, or members of another population evolving in the same problem domain, rather than by a fixed evaluation function [2].

Coevolution provides certain benefits over fixed evaluation evolution. Fixed evaluation functions are fine for evolving networks to approximate a known function such as XOR, but multiplayer games pose different problems to simple function approximation. Firstly, by what benchmark do we judge our players? Do skill levels at a given game approach a limit, or can an expert player always learn something to become better than other experts? The evaluation function in effect becomes the teaching force in the algorithm, examining its students and demanding that they score higher in the given test material. If a teacher is of a low calibre, his students, having surpassed him in skill, will reach a permanent plateau in their skill level, being able to satisfy their examiner 100% of the time and no longer receiving any pressure to become better. If on the other hand a teacher is of too high a calibre, he will be unfit to teach beginner

students who will have no idea how to begin to pass his tests, and thus will not be distinguishable from one another as good or bad students.

It is obvious that graded levels of examination are necessary to guide students from complete ignorance to steadily higher skill levels. Coevolution provides what is known as incremental evolution [20], in which solutions to large and complex problems are solved in portions of gradually increasing difficulty. Because testers for fitness evaluation are always taken from an evolving population in the same domain, testing difficulty is maintained at a level appropriate to the learners at all times.

A second problem is that multiplayer games often consist of multiple objectives to be solved in order to be a successful player. Because the challenge in a multiplayer game is set by the current opponent, being a good player means being able to defeat a wide variety of other players with different strategic strengths. For example, in backgammon one opponent may be particularly good at defensive strategies while another may focus purely on offensive strategies. These differing opponents are considered different objectives of the game, and a skilful player knows how to solve each different objective by having superior strategies in each case. Coevolution is able to provide a set of opponents of varying skill that test on a variety of objectives per fitness evaluation, training players to solve multiple objectives. This means that coevolution is more suited to multiplayer games than using a fixed evaluation function.

Thus, coevolution allows a population the ability to "bootstrap" itself up from beginner to expert level, by ideally maintaining a diverse set of test opponents, or *teaching set*, at an appropriate yet ever-increasing challenge level - without the need for any human expert knowledge.

There are however many typical problems involved in achieving successful coevolution. Cycles known as *intransitivities* can appear whereby individual A beats individual B, and B beats individual C, but C beats A, causing the population to cycle through different strategies without making further progress. This is possible in coevolution as the evaluation criteria is constantly changing, causing learners to focus on different criteria over time and making it possible to forget earlier learnt strategies [3]. Also, members of the population with weak overall skill but which pose interesting challenges in specific strategic areas may not survive long enough to encourage growth in those areas, thus leading learners to miss some evaluation criteria, a condition known as *focusing* [3]. More advanced algorithms for coevolution, including *fitness sharing* and *hall of fame* techniques, are used to work around these issues and promote higher quality evolution, and are discussed further in Chapter 4.

3 Backgammon

Backgammon is an ancient game of skill and luck, in which two players compete to be the first to bear all of their pieces from the board. Backgammon is at least one thousand years older than chess [18], with ancestral roots in ancient Mesopotamia and Persia.

Backgammon is known as a *Tables Game*, played on a board divided into 4 quadrants each with 6 long triangles, known as points, numbered 1 to 24 around the board on which the pieces of each player are set up symmetrically. The players take turns rolling two dice and moving their pieces around the board in opposite directions. Game play is made significantly more complex and interesting through offensive and defensive tactics – first, it is possible to land on a solitary opponent piece and send it back to the far end of the board, which must be re-entered on the board before play by the opponent can recommence. Second, pieces may be stacked on any point and left to prevent the opponent from moving, as no player may occupy a point already occupied by two or more enemy pieces.

Thus the game contains two sub-games - one in which the players' pieces are scattered and may race, attack or defend, and one in which all pieces have been moved such that attack is no longer possible, corresponding to a simpler race-state game.

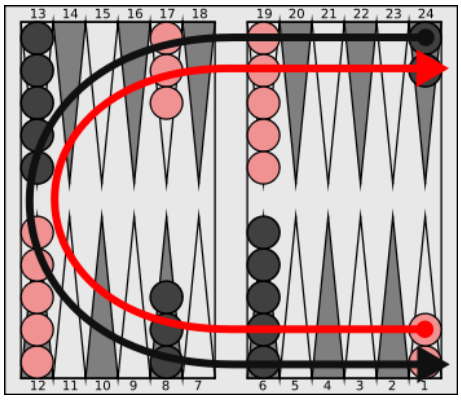


Figure 3.1: Backgammon layout and direction of play

3.1 Rules of the Game

The initial setup of the game is as in Figure 3.1; each player has two pieces on his 24-point, three on his 8-point, and five on his 13- and 6-points. Pieces are traditionally coloured black and white or black and red. Players move in opposite directions from point 24 to point 1, and must move all pieces first into their home quadrant, points 6 to 1 inclusive, before they may begin moving pieces off the board [9].

Play alternates with each player rolling two dice at the beginning of his turn. Upon rolling, a player moves his pieces according to the numbers on the two dice, moving one piece for each die. The same piece may be moved for both dice values but must be moved for each die separately. For example, if a dice roll shows 3 & 4 (denoted 3-4) one piece may move 3 and then 4, but not 7 all at once – if the opponent is blocking

the points for both the 3 and the 4 moves then that final 7 move cannot be made. If a player rolls doubles of any number, e.g. 3-3, that player must make four moves of that number. If any moves cannot be made, the player must move as much as possible. So for a 3-4 roll, the player must move the 4 if possible, otherwise just the 3.

Players can block each other from making a move by forming walls of two or more pieces on any point. No player may occupy such a point blocked off by their opponent, and thus it no longer forms part of the legal moves for that turn, even if the dice dictate movement to that point.

If a piece is sitting solitary on any point, it is vulnerable to attack. In the case that a player lands one or more pieces on a solitary piece of his opponent, the opponent must move that piece to the bar in the middle of the board. Before any further moves may be made by that opponent, he must place that piece back on the board during his next turn, requiring a die roll allowing re-entrance onto a point not already blocked by his enemy. So, a dice roll of 3-4 would allow that piece to return to either point 22 or 21, following which the remaining die roll may be played as usual. A player may not make any other moves unless all his pieces are off the bar.

3.2 Technical Details

Further to the basic rules are several technical points concerned with more serious tournament play and gambling.

If a player wins the game while his opponent has yet to bear off any pieces, the win is known as a *gammon* and counts for two wins/losses respectively. If the opponent has any pieces on the bar or still in the quadrant of points 24 to 19 when the other player wins, that win is called a *backgammon* and constitutes a triple win/loss respectively [9]. Backgammons are extremely rare in practice [18].

In addition to the normal game rules, a "doubling cube" is often used. This is a cube with the numbers 2, 4, 8, 16, 32, and 64 on it, which at the start of the game is placed in the centre of the board. Before rolling the dice, a player whose turn it is may propose to double the stakes of the current game, whereby the opponent can either accept or resign the game. If accepted, the opponent sets the cube down with the current stakes value face-up, keeping it until he decides to double again.

In major tournament play various extra technical rules and details have been used, none of which are relevant to this work.

3.3 Strategy

Backgammon has a well-established theory of move strategies generally employed by more advanced players, including a *running game*, *priming game*, and *duplication* (in ascending order of complexity) [5]. A running game involves trying to move as quickly as possible to the end of the board. A priming game involves building consecutive obstructing walls, known as primes, to impede the opponent's pieces trapped behind that wall. A wall covering 6 consecutive points cannot be passed by any opponent pieces. Duplication involves placing one's pieces in order to limit the usefulness of the dice to the opponent, e.g. by positioning pieces such that the opponent has to roll a 2 to hit any of them.

3.4 Artificial Intelligence in Backgammon

The game of backgammon has been used for many years as a tool in the study of AI. Backgammon poses an interesting challenge for AI, as it requires great levels of skill and sophistication to play at an expert level, yet at the same time is impossible to know for sure who will win the game at most given moments of play, due to the probabilistic element introduced through dice rolls.

Early attempts at backgammon learning programs used evaluation functions with large numbers of hand-crafted features based on expert human knowledge. In 1977 Hans Berliner created BKG, a static evaluation function created by hand without the use of any machine learning techniques [1, 15]. Despite being hand-made, BKG proved that human expertise at backgammon could be expressed using static evaluation functions. Then in 1987 Neurogammon was presented by Tesauro & Sejnowski, which used the backpropagation algorithm to train multi-layered neural networks on training sets of move evaluations made by expert human players [15]. This network was a fair player and won the Computer Olympiad in backgammon in 1989 [16], but did not play at a master's level. Following this, Tesauro published TD-Gammon in 1992 [18] which trained neural networks using temporal difference (TD) learning, a learning method based on updating the value estimate of a current move based on expected returns, and self-play, whereby a single move-evaluating network is trained by playing itself at many games of backgammon. By increasing the number of hidden layers in TD-Gammon's networks, implementing certain expert knowledge features into the system, and running for longer training periods, Tesauro was able to create a formidable backgammon player that not only came close to defeating top masters of the game, but demonstrated superior strategies not previously understood or valued by human experts.

Following Tesauro's work in TD-Gammon, Pollack & Blair presented a paper claiming that Tesauro's success in backgammon with self-play learning was not as earth-shattering as it appeared, due to their results that a simple naïve hill-climbing algorithm could come close to achieving similar results to TD-Gammon [10]. They argued that the success of TD-learning and hill-climbing came more from the basic dynamics of the backgammon domain and learning environment than the self-play learning algorithm itself.

Tesauro later responded in kind to Pollack & Blair [19], pointing out several weaknesses in their argument. First, he argued that the relative difference in benchmarked skill-levels of hill-climbing versus TD self-play was more significant than Pollack & Blair had assumed, resembling the difference between an average human player and a world class champion player. Second, he argued that this weakness in the hill-climbing approach is due to an inability to extract nonlinear solutions, despite the existence of hidden nodes in their neural network structures.

Following this clash came the first work in coevolution in the backgammon domain, by Paul Darwen [2]. Darwen compared coevolution to Tesauro's TD-learning, and approached the backgammon learning problem in two stages, first attempting to coevolve for the purely linear case of a network with 0 hidden nodes, and then attempting to coevolve more complex nonlinear solutions for structures including hidden nodes. He discovered that by using a population of 200 individuals and very long training

times (in the order of 80 million games, compared with TD-Gammon's 1.5 million) he could evolve networks to a plateau slightly surpassing TD-learning for the linear case. However, his work on nonlinear networks did no better than the linear case, and his subsequent analysis of network weights showed that indeed, no nonlinear structure was being evolved. Darwen states that nonlinear solutions may require infeasibly large numbers of games to learn the same skills as TD-Gammon, due to the "all-or-nothing death-or-survival" [2] approach of coevolution, and the vastly larger weight search space caused by hidden layers.

This previous research provides the background for this thesis. TD-Gammon proves that neural networks can be trained to play backgammon at a master level. Pollack & Blair's results indicate that the backgammon domain is ideal for coevolutionary learning, although suffers from apparent intransitivities [10, Section 3.3]. Their hillclimbing algorithm also obtains a skill plateau considerably lower than that of coevolution or TD-learning [2, 18]. Darwen's results indicate that coevolution is useful in the backgammon domain and does very well for linear network structures, but cannot be used to learn nonlinear structure, possibly due to computational limitations.

3.5 Using Neural Networks for Backgammon Play

Neural networks used to play a game of backgammon typically take up the function of "move evaluators", whereby their input state is a representation of a state of the game of backgammon, and their output a value proportional to the chance of winning the game from that game state. More complex representations such as Tesauro's TD-Gammon networks use up to five output values, one showing the probability of winning the game, two showing the probabilities of winning and losing a gammon, and another two showing probabilities of winning and losing a backgammon.

Each time the network chooses a move a list of legal moves is made based on the game state, the rules of play, and the current roll of the dice. Each move is considered in turn by evaluating the state of the game *after* that move would be made, and assigning each move in the list a score, being a combination of the networks probability of winning from that state, and its probabilities of winning/losing a gammon or backgammon. Finally the move with the best score is chosen, and it is the next player's turn. This is possible in the backgammon domain because we have a partial model of the game – although we don't know what the game state will be for the beginning of our next turn, we do know exactly what state the game will be in after we make our current move (and before the opponent makes his next move). This in-between state is known as an *afterstate* [14]. The best neural network to play backgammon is the one most accurately able to predict its chances of winning from any afterstate, and therefore make moves which maximise the chance of winning throughout the entire game.

4 Coevolution for Backgammon

We begin by asking whether coevolution is helpful in training backgammon players. The level of success of Pollack and Blair's hill-climbing algorithm [10] is surprising as hill-climbing is a pared down form of evolution with a population size of just 2, raising the hypothesis that true evolution with a larger population will not provide any benefit. We test this by comparing the results of using different population sizes for coevolution. We also test to see if the incremental evolution provided by coevolution is necessary, by comparing coevolution to evolution using a fixed fitness evaluation. Then we use fitness sharing and hall of fame coevolution techniques to try to achieve better results than those attained by the basic round robin tournament as used by Darwen [2].

All experiments in this work use the *NeuroEvolution of Augmenting Topologies* (NEAT) algorithm for evolution, presented by Stanley & Miikkulainen [13]. This algorithm was developed for evolving network topologies as well as weights, however in this work topological mutations were switched off and only weights were optimised. NEAT was first used in this thesis for investigating topological as well as weight optimisation, however early results were not promising and topological optimisation was abandoned in favour of other lines of investigation. Algorithm parameters used in this work are presented in Appendix 8.2.

To provide some metric of how successful our backgammon players are, the benchmark player Pubeval was used. Pubeval is a linear backgammon move evaluator function created by Tesauro, trained on a lexicon of expert human backgammon knowledge and released to public domain in 1993 [17]. Pubeval plays at an intermediate human level and has been used as a benchmark by many backgammon learning programs, including those of Darwen [2], Pollack & Blair [10], and Tesauro [18]. Pubeval is thus ideal for benchmarking our work in order to compare experimental results to each other as well as to others' works. Benchmarking against Pubeval involves periodically sampling a champion from the evolving population, and using it to play a number of games of backgammon against Pubeval. These scores are then graphed to give an external view on how evolution is proceeding – the score of these games has no bearing on fitness values, and does not change the evolutionary process at all.

4.1 Population Size Comparisons

Population size affects coevolutionary learning strategies in two ways. A larger population size can mean a broader range of different teaching set opponents for an individual to be tested against, and it can mean a wider search as the algorithm moves through the search space. Pollack & Blair's results indicate that fair backgammon players can be trained using a basic hill-climbing algorithm, which is a form of evolution with a population size of 2, and thus a search width of just one solution. We investigate whether search width is important for optimisation in the backgammon domain by comparing evolution with varying population sizes.

For this purpose we use a full *round robin* tournament coevolution strategy, as used by Darwen [2]. In round robin coevolution, individuals are evaluated against all other individuals in the same population, and their final fitness score is the average

score received. For backgammon, these scores are the result of playing a number of backgammon games against each of the other individuals; an average score represents the proportion of games won by that individual.

Modifications were made to the algorithm for steady-state evolution. Initially a normal round robin tournament is played amongst all initial population members to calculate their fitness values. Then, each time a new individual is evolved it is tested against all other existing members and the results of each game are used to calculate the new fitness value of both players. Furthermore, fitness values must only be taken from scores achieved against individuals *still in the population*, meaning when an individual has been removed its score history is no longer useful to those that played it. To this end, an $N \times N$ matrix, where N is the population size, is maintained with scores between individuals. Every time an individual is removed, its replacement is assigned the same matrix indices, and its entries updated to reflect the scores against the new individual. Then, at the end of every round, each population member's fitness is recalculated from the matrix.

4.2 Fixed Evaluation vs. Coevolution

Incremental evolution, provided by coevolution, provides evolution with fitness evaluation criteria that evolve along with the skill level of the population. In order to test whether incremental evolution is useful for training backgammon players, we compare coevolution to evolution with a fixed evaluation criterion.

For fixed fitness evaluation the benchmark player Pubeval was used as the fitness evaluator. The fixed evaluation test involves playing each individual against Pubeval for a series of backgammon games, averaging the scores to get a fitness value between 0 and 1.

For comparison to coevolution the resulting players were also externally benchmarked using Pubeval. This is different to using Pubeval as a fitness evaluator. As a fitness evaluator, Pubeval is an active part of evolution, and the scores against Pubeval are used directly as fitness values. However, during benchmarking, a champion network plays Pubeval for a larger number of games simply to ascertain its score against Pubeval, which does not affect fitness values or evolution in any way.

4.3 Coevolutionary Strategies

Coevolution entails a constantly changing set of evaluators, which can cause intransitivities as described in Section 2.4. Pollack & Blair demonstrate the existence of intransitivities between later generational champions [10, Figure 5] which may be preventing coevolution from further optimisation in the backgammon domain.

In order to deal with the presence of intransitivities it is helpful to maintain a diverse set of opponents for fitness evaluation. This helps to prevent the changing evaluation criteria from focussing too much on particularly successful strategies, reducing the probability of cyclic behaviour [3]. Rosin & Belew [11] present methods for maintaining diverse sets of opponents in coevolution and thus better cope with the intransitivities seen in the domain, so we compare the use of two of their methods to the round robin strategy already used, to see if they train better backgammon players.

The coevolutionary algorithms compared are single-population round robin tournament coevolution, double-population *fitness sharing* coevolution, and fitness sharing using a *hall of fame*.

4.3.1 Fitness Sharing

Fitness sharing coevolution involves two genetically distinct competing populations for coevolution, each population being used to evaluate the other. However, rather than simply using the average score or "simple fitness" value as in round robin coevolution, fitness sharing aims to take into account similarities of individuals within a population. An individual is rewarded if it is able to beat an opponent from the other population that few others can. Likewise, if an individual beats an opponent that everyone else in the population also beats, then that score does not contribute as much to its final fitness value. This way, the teaching set of opponents is diversified and important genetic innovations are more likely to be retained in the population [11].

Each round, a new individual's fitness is set to 0 and it is tested against each member of the opposing population. For each opponent j that it defeats, its fitness value is incremented by $\frac{1}{N_j}$, where N_j is the number of individuals from the same population also able to defeat opponent j. So the shared fitness for an individual who manages to defeat opponents with the set of indices X is:

$$\sum_{j \in X} \frac{1}{N_j}$$

Fitness sharing for steady-state evolution requires a slight algorithmic modification, as with the round robin strategy. First, the two initial populations play a normal fitness sharing tournament against each other. Then, for each population an individual is bred and tested against the other population, storing scores as $\frac{1}{N_j}$ in an N by N matrix similar to steady-state round robin, where both populations are of size N. All fitness values are then re-evaluated from the matrix at the end of the evolutionary round.

4.3.2 Hall of Fame

One of the main problems with coevolution is a phenomenon known as *coevolutionary forgetting* [4]. Because coevolution deals with finite population sizes, often individuals from past generations who provide good evaluation criteria are lost and have to be rediscovered again later. This can cause cyclic effects in strategy learning, slowing or stopping the coevolution process. In order to prevent forgetting it may be necessary to use a *coevolutionary memory* [8] – the hall of fame (HoF) being one such tool.

A HoF is simply a list of past generation champions – in the steady-state case, population champions sampled at even intervals. During evaluation, a sample of these past champions is used in addition to the tester population. This saves potentially useful genetic material for future generations to be tested against. A steady-state implementation incurs slightly more computational cost than regular evolution because every time a new HoF sample is taken, all individuals in the current populations must be tested against the new HoF sample, maintaining identical teaching sets for all population members.

4.4 Experimental Setup

For the neural networks to efficiently represent backgammon move selectors, we use a basic linear version of the representation used by Tesauro's TD-Gammon [18], with 198 input nodes, no hidden nodes, and 1 output node representing probability of winning. The input nodes are a hand-crafted representation of the board state, describing the current positions of all the player's and opponent's pieces on the board, off the board, and on the bar.

Throughout all experiments the simplest version of backgammon is used. No doubling cubes are considered, nor is a gammon or backgammon rewarded or penalised. Games are played to the end, whereby the winner receives 1 point, the loser 0. Because evaluations are based on the results of multiple games, an individual's final fitness score is averaged to always be a value between 0 and 1 indicating proportion of games won.

The population test uses full round-robin strategy with population sizes 2, 3, 5, 15, and 141, with respectively 140, 70, 35, 10, and 1 game(s) per opponent in order to have a total of 140 games per fitness evaluation in all 5 cases. Each test in this experiment was run to a total of 8 million games.

The fixed evaluation test was run using 15 population size and 140 games per evaluation against Pubeval, also to 8 million games.

The experimental setup used to compare coevolution strategies of round robin, fitness sharing, and HoF involves steady-state coevolution with populations of size 15, using 11 games per opponent[3], and running for a total of 10 million games per experiment. For the fitness sharing plus hall of fame model, a hall of fame with a maximum sample size of 5 is used to supplement the size 15 population.

All experiments were run 10 times and averaged to get the final graphed results and confidence intervals.

For all benchmarking, Pubeval was used to test champion networks sampled every 200 evaluations. The score against Pubeval is averaged over 1000 games.

[3]Odd numbers are used for games-per-opponent parameters in most of the experiments in this thesis. This is simply a nicety making it impossible to have any tie results of 50%.

4.5 Results

The results of the population test can be seen in Figure 4.1. By raising the population size the skill plateau becomes noticeably higher until size 15. After 15, increasing the population size has less impact on the plateau, at least within the 8 million game period played here. Population size 141 appears to still be learning after 8 million games. 95% confidence intervals for these results can be seen in Table 1, sampled every 2 million games. Larger populations are far more consistent than lower populations, with confidence intervals greater than 50% of the mean for a population of 2, and only 5% for a population of 141.

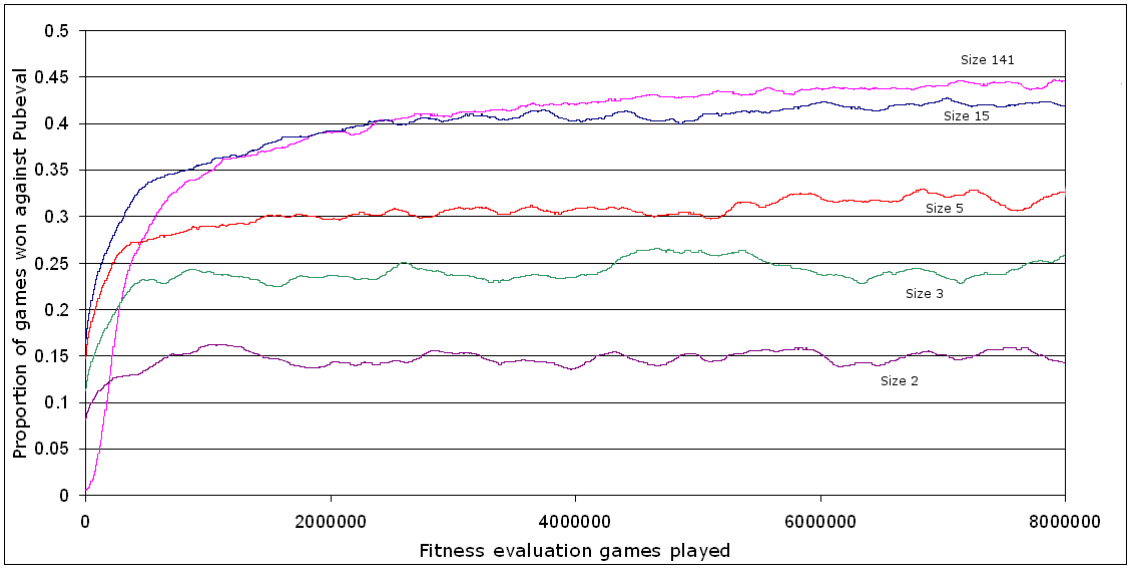


Figure 4.1: Population size test comparing population sizes 2, 3, 5, 15 and 141.

Population Size	2	3	5	15	141
2×10^6 games	68.5%	27.4%	11%	9.1%	5.7%
4×10^6 games	54%	22.1%	15%	10.4%	4.8%
6×10^6 games	45.8%	20.4%	17.3%	8.1%	5.1%
8×10^6 games	46.8%	19.8%	11.7%	7.8%	5%

Table 1: 95% confidence intervals for the population size test sampled at 2, 4, 6 and 8 million games. Confidence intervals are expressed as percentage of the mean value at the sampling point.

Figure 4.1 demonstrates that population size is important for optimisation in the backgammon domain, and therefore that evolution with a sufficient population size is more effective than hill-climbing. This is encouraging, as Pollack & Blair successfully trained backgammon players using a hill-climbing algorithm, achieving a plateau of 0.4 against Pubeval [10].

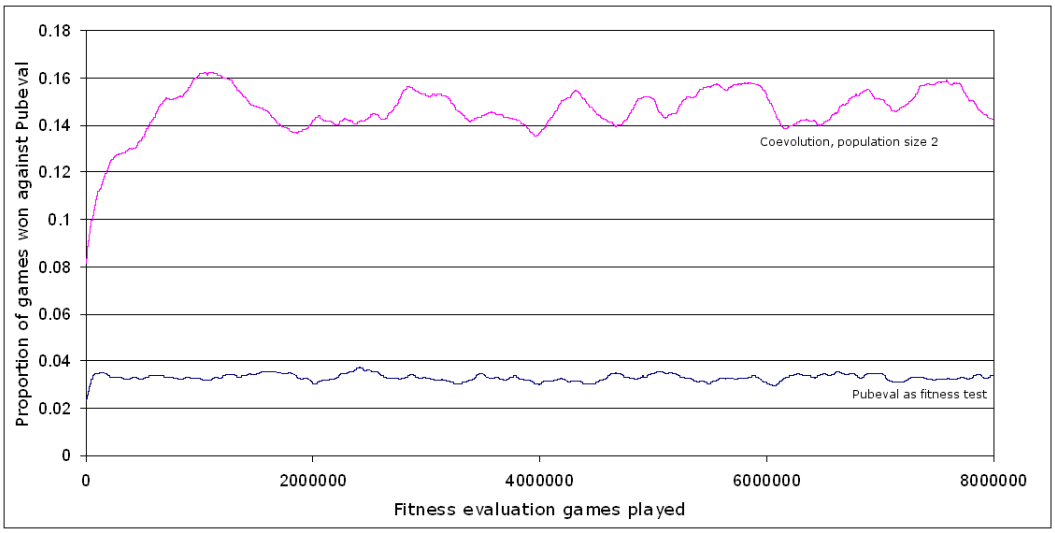


Figure 4.2: Comparison of fixed evaluation evolution using Pubeval and the dynamic teacher selection of coevolution.

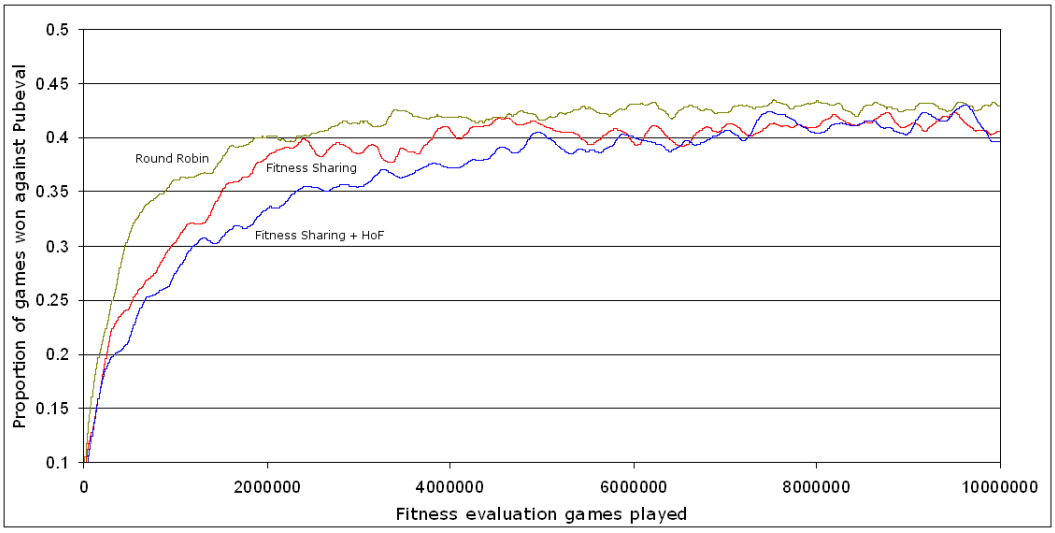


Figure 4.3: Comparison of round robin, fitness sharing, and fitness sharing + HoF.

Figure 4.2 shows evolution using Pubeval as a fixed fitness evaluation. We compare to the least successful results for coevolution obtained in the population test, demonstrating that it performs far worse than all coevolution results obtained so far, plateauing immediately at just under 0.04. 95% confidence intervals for the fixed evaluation

strategy were between 90-105% of the mean, with a maximum score of 0.079 during benchmarking. These results show that we need coevolution to provide incremental evolution for evolving backgammon players – by using fitness tests of steadily increasing difficulty coevolution far outperforms evolution, which gets stuck very quickly on a low skill plateau.

Figure 4.3 shows the results of the coevolution strategy comparisons. The round robin approach was able to beat pubeval 40% of the time after 2 million games, getting to 43% after a further 8 million games. The fitness sharing and hall of fame models learn slightly slower at first, which is not surprising given the extra numbers of games played per generation. However, neither of these approaches succeed in scoring a higher score than the normal round robin tournament. 95% confidence intervals remain between 8-11% for all three strategies.

Pollack & Blair's results in [10] indicate the presence of intransitivities in the backgammon domain, and so we expect to achieve better results than round robin coevolution by using fitness sharing and hall of fame techniques, designed to diversify teaching sets to better cope with intransitivities. However, the results of Figure 4.3 demonstrate that there is in fact no improvement. We therefore go on in the next chapter to investigate the hypothesis that the backgammon domain is actually *not* intransitive, because there was no improvement to coevolution by using methods designed to deal with intransitivities.

5 Transitivity Analysis

The fitness sharing and hall of fame strategies are designed for coevolution in an intransitive domain [11], and given the evidence that intransitivities exist in the backgammon domain [10] we expect them to provide an improvement over simple round robin tournament coevolution. However, the results of Chapter 4 show that there is in fact no improvement. In this chapter we investigate the hypothesis that we found no improvement because the backgammon domain is transitive, and therefore that coevolution in the backgammon domain is not being impeded by intransitivities.

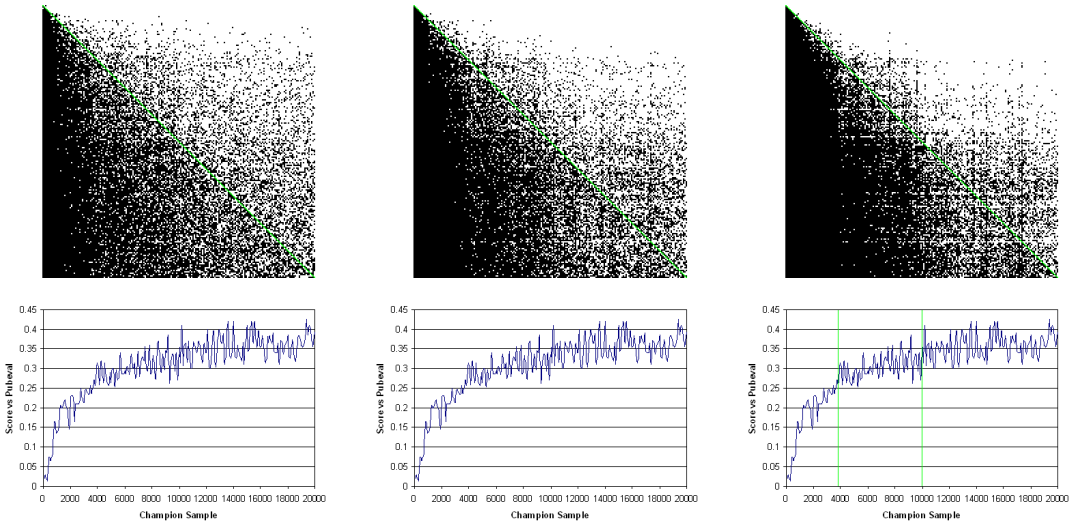
5.1 Champion Tournament Grid

In order to examine this hypothesis we need some way of inspecting the domain for evidence of intransitivities. Rosin & Belew [11] demonstrate the use of a grid displaying results of a tournament amongst generational champions in order to visualise champion progress during learning. Each generation the population champion is saved, and at the end of coevolution the champions are tested against all other champions in a full round robin tournament. A grid is set up with rows and columns corresponding to population champions going from left to right and top to bottom. Results of each contest are shown as a black dot if the row-champion won, or a white dot if it lost. As both columns and rows represent the same list of champions, this grid is symmetric about the diagonal. In this way it is possible to detect intransitivities. In transitive domains where every generation outperforms the previous, we should see a black triangle in the lower-left diagonal half of the grid, and a white triangle in the upper-right. In the presence of intransitivities the grid colouration should become mixed, with white dots in the lower-right black triangle and vice-versa.

We ran a further round robin experiment for 20,000 evaluations, saving champions periodically every 100 evaluations and later running a full round robin tournament between these champions.

Figure 1(a) shows the results of our first experiment using 30 games per match. Below this is the benchmarked result of each champion, played against Pubeval for 1000 games each. The grid colouration is very mixed, which indicates the presence of intransitivities. However, the backgammon domain is not deterministic – the luck of the dice can sometimes mean that a poor player beats a more advanced player, and thus results of these backgammon matches may be affected by noise. To investigate if this is the case, further experiments were run using higher numbers of games per match, as seen in Figures 5.1(b) and 5.1(c). We see by using 200 games per match that the grid resolves into a pattern of three sections in Figure 5.1(c) – a leftmost section of almost complete blackness indicating strictly improving players, followed by a section of light black and white mixing, and finally a triangle of heavy black and white mixing in the bottom right corner. Using more games per match than 200 ceases to have a noticeable effect on the grid's appearance.

The two vertical lines traced on the Pubeval results beneath Figure 5.1(c) demarcate these three sections. The steepest part of the learning curve corresponds almost exactly to the darkest section of the grid, following which comes a section of slower learning, and finally a noisy plateau in skill corresponding to the final mixed black and white



(a) 30 games per match (b) 90 games per match (c) 200 games per match

Figure 5.1: Round robin grids with different numbers of games per match. The high level of black and white mixing in (a) seems to indicate intransitivities, while (b) and (c) show by increasing the games per match that apparent early intransitivities are caused by noise in the fitness evaluations.

section of the grid.

Unfortunately a question still remains. The lower-right grid section of mixed black and white colouration could be demonstrating that the plateau is caused by intransitivities amongst later generations, whereby cycling strategies cause evolutionary progress to slow down or stop and therefore champions get beaten by previous generations. However, it is clear from Figure 5.1(c) that this black and white mixing is only occurring at the phase of evolution when benchmarked skill against Pubeval is not increasing very quickly. During such a phase of evolution, champions are clearly not outperforming their ancestors to a large degree, and we expect the outcome of a game between such similar individuals to be very unpredictable for a stochastic game such as backgammon. This would therefore provide such a mixed colouration to the grid, even in a purely transitive domain.

It is clear from Figure 5.1(c) that there are no intransitivities between champions in the first grid section, as shown by the solid black coloured left edge to the grid. However, it is still not clear whether intransitivities exist between later champions, during the last sections of evolution. We therefore need a final test to investigate intransitivities during periods of low evolutionary improvement.

5.2 Plateau Analysis

We devised a final experiment to investigate intransitivities between later generational champions. If there are intransitivities occurring, individual champions would still see an improvement over each other per generation. That is, for a skill plateau in a purely

transitive domain there will be no local or global improvements in skill, while for a plateau caused by cycling strategies in an intransitive domain we will still see constant local improvements between neighbouring generations. In order to test this, we first made each champion play the champion from the previous generation, playing 10,000 games per match to be certain of minimal interference due to the noise inherent in the fitness evaluation process. Then, each champion also played the champion from two generations previous. This way, if the domain is transitive, we will see the results of these matches converge to 0.5 (a tied match), with both first-previous champion and second-previous champion results following each other closely. If the plateau is caused by intransitivities however, we expect to see the first-previous results converge to a value slightly greater than 50%, and to find cycles within the data whereby champion A defeats champion B, B defeats C, and C defeats A. Figure 5.2 shows the results.

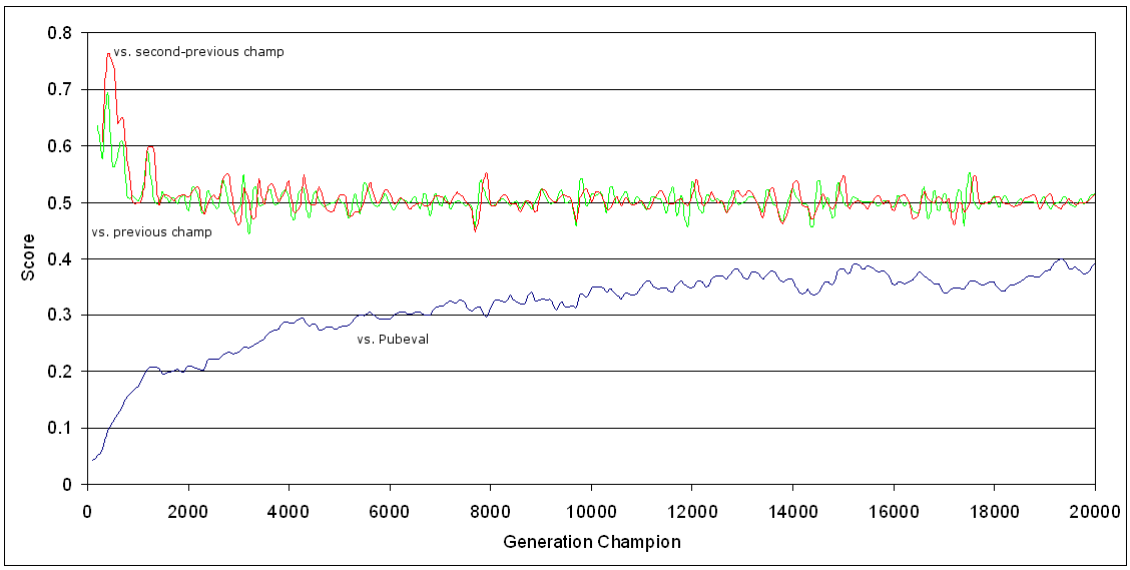


Figure 5.2: First-previous and second-previous generation champion tests.

The results of these matches do indeed converge to 50%, and both first-previous and second-previous tests follow each other very closely. A deeper inspection was made of the raw data to search for cycles that may not have shown up on the graph. These could be identified by first scanning for an individual A that beats the first-previous champion B but loses to the second-previous champion C, and then examining whether B beats C or not. If cycles with very small win/loss margins in the order of 1-2% are present, they may not be clearly visible on the graph. However, no cycles at all can be identified, even with such small win/loss margins.

There is now strong evidence indicating that there are no intransitivities amongst generational champions of coevolution in the backgammon domain. Though Pollack & Blair demonstrated apparent cycles of expertise in later champions [10], it is clear from Figure 5.1 that with too few games per match, results are unreliably noisy. As their work involved at most 8 games per fitness evaluation, it is clear that their observed cycles are not the result of intransitivities, but noise effects corrupting the optimisation process. This explains why fitness sharing and hall of fame techniques provided no

benefit to coevolution of backgammon players, because these methods only improve coevolution in the presence of intransitivities.

We see in Chapter 4 that coevolution benefits from having a large population size, however we know now that the backgammon domain is strongly transitive, meaning that maintaining a diverse teaching set of opponents is not important to successful evolution. This shows that in the backgammon domain, population size is important to coevolution for a broad search, but not for a diverse teaching set, as described in Section 2.4.

These results also help to illustrate Tesauro's TD-Gammon success [18]. Tesauro used self-play to train his networks, whereby a network learns from games played against itself, meaning the population size is effectively just 1. Although Tesauro used reinforcement learning rather than evolutionary computation, the transitivity of the backgammon domain still helps explain why the self-play technique works so well.

Thus we are left with an interesting challenge: can this domain transitivity be exploited for better coevolution strategies, taking less computational effort and outperforming the limitations discovered by Darwen [2] for optimising nonlinear network structures?

6 Efficient Evaluation in Transitive Games

In order to exploit the domain transitivity discovered in the previous chapter, three new fitness evaluation strategies are proposed. "Binary rank placement" exploits rank transitivity using a binary search to play only a subset of the population during fitness evaluations. "Single evaluator" uses a network from the evolving population to test all other networks, with their respective fitnesses based purely on that one match. Single evaluator is based on the hypothesis that, as the domain is transitive, a diverse teaching set is unnecessary for coevolution. Finally we analyse the probability distribution of new individuals' fitness rankings during the round robin scheme, and discover that 50% of the time a new individual is ranked worst, and thus is unhelpful to evolution. This motivates the "losers first" scheme that tests a new individual first against the worst player in the population, aborting the fitness evaluation if the match is lost and discarding the newcomer immediately.

In this chapter we compare these strategies to the initial round robin results, and look at the effects of changing the number of games played per evaluation.

6.1 Binary Rank Placement

The binary rank placement scheme uses a binary search to find a new individual's rank within the population. Because of the evidence for domain transitivity we assume that the ranking order of a population is strictly acyclic in the sense that if a newcomer beats some population member, it is certainly superior to all lower ranked members as well.

In the first round, a full round-robin tournament is played between all individuals to get initial fitness values. These values are translated into the ranked position of each population member, from 1 to population size N. Whenever a new individual is introduced to the population it begins by playing a match against the middle-ranked individual, and the population is divided into two sub-populations. Should the newcomer win its match, it goes on to play the middle-ranked individual from the upper (higher fitness valued) half-population, otherwise it plays that from the lower half-population. This process is incrementally repeated until the newcomer's correct position in the ranking order has been discovered, either by losing to the worst player, winning to the best player, or by having won and lost to a neighbouring combination of players. Any networks ranked lower than the newcomer move down one rank, and the worst individual is removed in preparation for the next generation.

This form of binary search thus ensures a newcomer only need play $O(\log(N))$ opponents in its fitness evaluation, instead of $O(N)$.

6.2 Single Evaluator

Coevolution provides both incremental evolution and potentially diverse teaching sets to the evolutionary process, as described in Section 2.4. Single evaluator ranking makes the extreme assumption that, as the domain is highly transitive, it is possible to get an accurate fitness evaluation of an individual based on a single match against one

opponent, rather than a potentially diverse set of opponents – provided incremental evolution still occurs.

Single evaluator starts with a full round-robin tournament to ascertain fitness values of the initial population. The best individual is promoted to "teacher", and fitness values of all members of the population are set to be their score against this individual. Thereafter, each newcomer plays the teacher for one match to get its fitness score.

Because incremental evolution is important for evolving in the backgammon domain, this teacher network must be upgraded consistently to allow coevolution to proceed. Thus, when a teacher is defeated by some proportion of games k, it is replaced by the individual that defeated it. When a teacher is replaced, all population members must be retested against the new teacher in order for their fitness values to correctly relate to newcomers' fitness values. This is an extra and non-optimal computational cost, as these extra games do not add anything to the knowledge of the system. Also, hand-picking this parameter k places responsibility for effective incremental evolution on the experimenter, rather than allowing coevolution to automate it.

6.3 Losers First

Finally, we investigate the distribution of the rank of new individuals to the population, combining this with the knowledge of domain transitivity to devise our third fitness evaluation strategy, losers first.

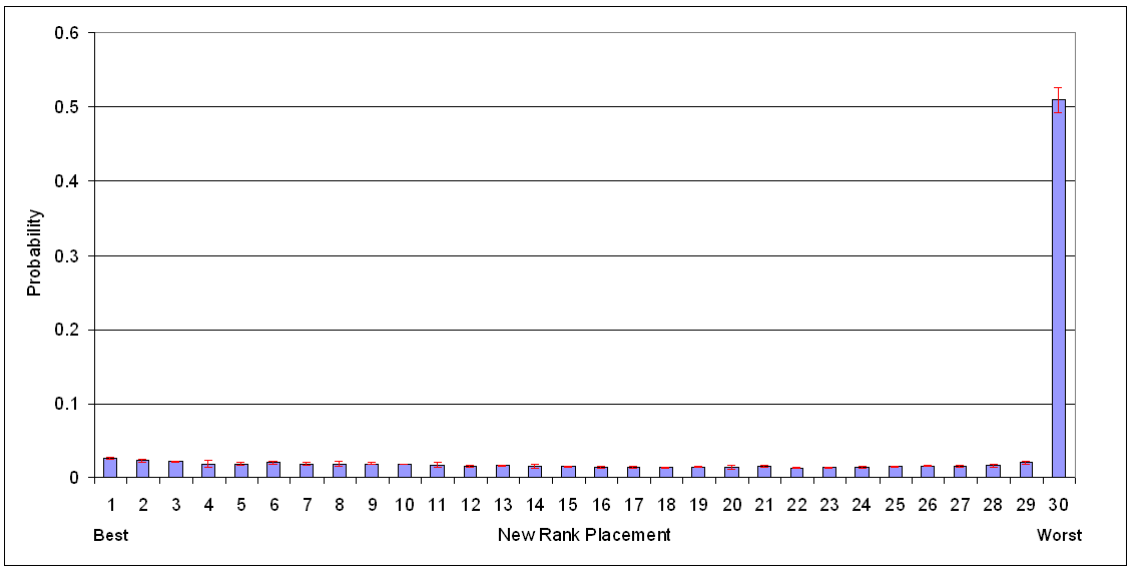


Figure 6.1: Distribution of rank placements of new individuals.

To calculate this distribution a series of 5 full round robin experiments was run, each with a population size of 30 and a large value of 101 games per match to reduce noise in the rank placements. Each experiment was run for 50,000 evaluations. The results can be seen in Figure 6.1, with 95% confidence intervals. We can see that, while ranks from best to second-worst have quite an even distribution of around 0.02, 50% of the time the new individual ranks worst. This reflects the danger of using random

mutations in evolution. Clearly, half of the time these mutations are disastrous, driving a newcomer below the skill level of the worst population member.

This suggests that we can avoid a lot of wasted games by first ascertaining if a new individual is the worst, and if so aborting further evaluation and removing the individual from the population ready for the next evaluation.

In the losers first scheme, each time a new individual is introduced it plays its first match against the worst ranked population member. If it loses, it is immediately removed from the population and a new individual is generated and tested. This is reasonable due to the evidence of transitivity, because if an individual is worse than the worst ranked network in a population, it is also worse than the other members of that population. If the individual wins however, it continues playing a full round robin tournament against the remaining population members in order to calculate its fitness value.

Note that the losers first form of fitness evaluation is only possible using steady-state evolution, whereby the worst member of the population can be identified before each fitness test. (In generational evolution, all individuals in a population are tested in parallel.)

6.4 Results

Before testing the single evaluator strategy, we compare it with different values of k, the score at which a teacher is replaced. Values of 0.65, 0.7, and 0.75 are compared as these seem intuitively good choices for k, neither too close to a draw at 0.5, nor too difficult to achieve. The results of this test can be seen in Figure 6.2.

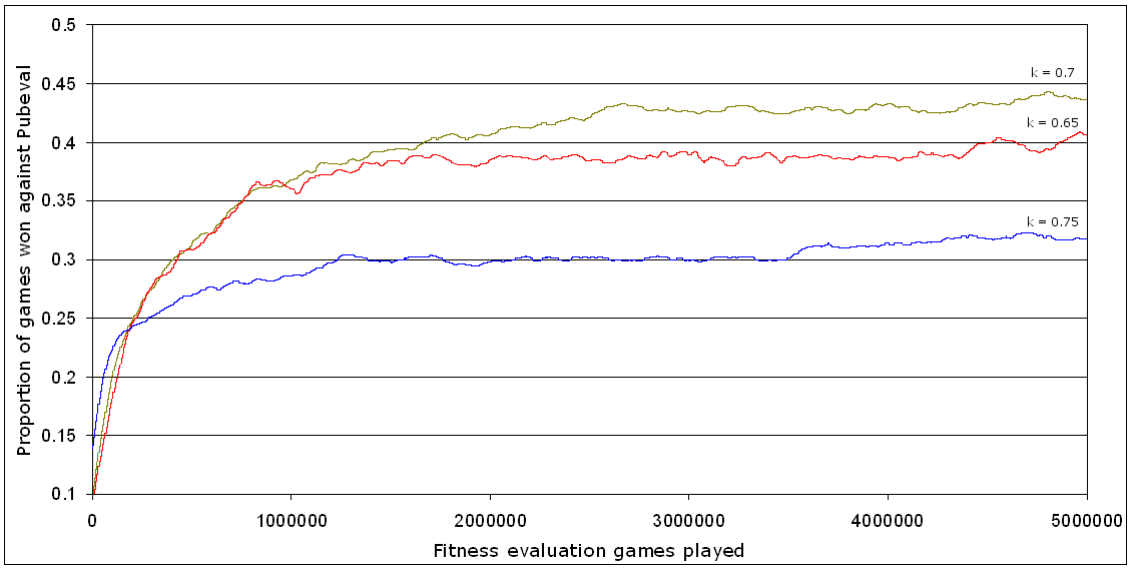


Figure 6.2: Comparison of k values for the single evaluator scheme, demonstrating parameter sensitivity. These experiments use 30 population size and 51 games per opponent.

These results indicate that 0.7 is a good choice for the following experiments. Performance is worse for a k of 0.65, as this causes the teacher to be replaced too often, which each time necessitates rematches and wastes games. 0.75 is apparently too high and doesn't replace the teacher often enough, causing a low skill plateau in the same way as the fixed evaluator results in Figure 4.2.

Before testing the binary rank placement and single evaluator schemes, it is necessary to investigate the effect of using different numbers of games per opponent for fitness evaluations. The reason for this is that both schemes use a significantly lower number of opponents per fitness evaluation. For example, with a population size of 15 and 11 games per opponent, round robin and losers first use 154 games per complete fitness evaluation, single evaluator uses just 11, and binary placement uses a maximum of 44. Results in Chapter 5 demonstrate that noise is a fundamental issue with backgammon fitness evaluations, and thus using more games per opponent may be necessary.

First we examine the binary rank placement scheme, testing it at 15 population size using 11, 41, 81, and 101 games-per-match.

As seen in Figure 6.3, raising the number of games per opponent gives an improvement in the plateau level with 101 games per opponent giving a score of around 42% by 10 million games, similar to the round robin scheme. Note that in this and following graphs in this chapter, we refer to population size and games-per-opponent

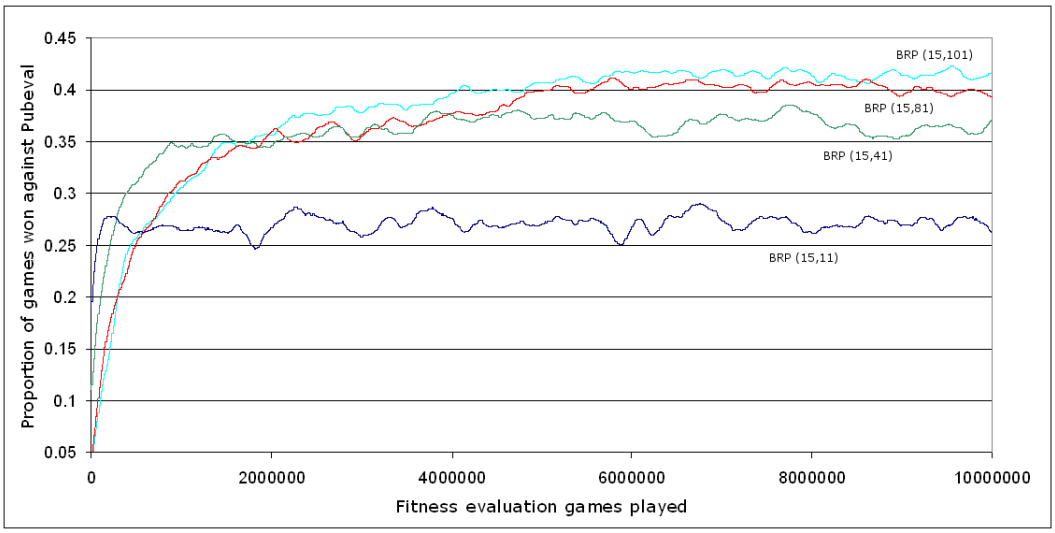


Figure 6.3: Comparison of games-per-match for the binary rank placement scheme.

parameter settings using the form (P, G), where P = population size and G = games-per-opponent.

Following this the single evaluator scheme was tested, also using 15 population size, with 11, 31, 41, and 81 games-per-match.

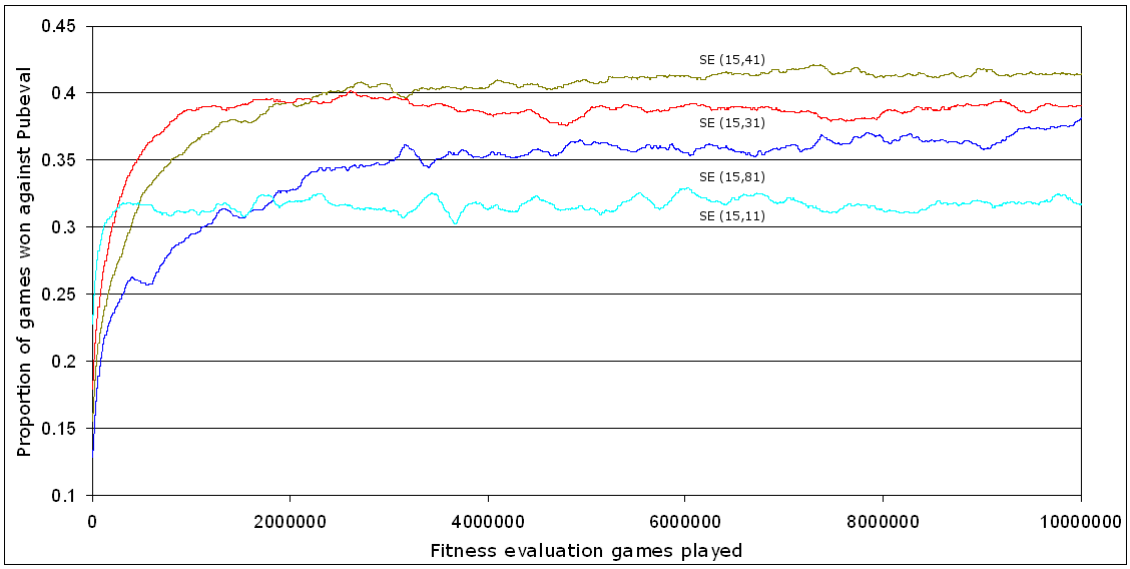


Figure 6.4: Comparison of games-per-match for the single evaluator scheme.

Figure 6.4 shows that increasing the games per opponent to 41 raises the skill plateau, however by increasing it further to 81 we see a decrease in skill level. This is because extra unnecessary games are being played per match, making the algorithm slower. So there appears to be an optimal choice of games per opponent, trading off between speed and correctness of fitness evaluation.

Finally, the comparison of losers first, round robin, single evaluator and binary rank placement can be seen in Figure 6.5.

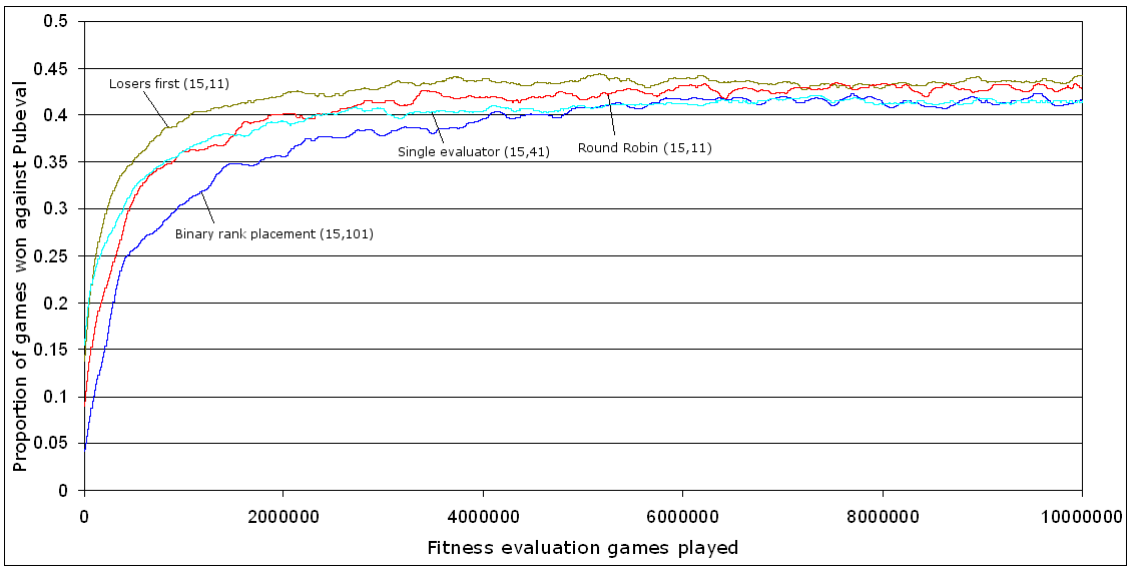


Figure 6.5: Comparison of losers first, round robin, single evaluator and binary rank placement.

The losers first strategy provides a clear gain in optimisation speed. It plateaus at a slightly higher level than round robin, achieving almost 45% against Pubeval, and it reaches that plateau much faster.

Binary rank placement fares the worst, with a slower learning curve and a plateau equal to the single evaluator scheme. It is unable to surpass round robin however. This indicates that the gain achieved by playing fewer opponents is outweighed by the extra games necessary for correct evaluation per opponent.

The single evaluator scheme optimises comparably well to the round robin scheme, starting with a steeper learning curve but soon tailing off to achieve a plateau of 1-2% worse than round robin. This is due to the k parameter introduced by the single evaluator scheme. We see from Figure 6.2 that results are sensitive to this parameter, and Chapter 4 demonstrates that incremental evolution provided by coevolution is very helpful to the optimisation process (Figure 4.2). By hand-picking the criteria for incremental evolution, rather than allowing coevolution to provide it automatically, we are introducing a source of error to the system, causing enough loss of accuracy to render this strategy less efficient than round robin.

6.5 Analysis

The binary rank placement results in Figure 6.3 show a major problem with binary rank placement. The stochastic nature of backgammon, which can in rare cases allow a beginner to beat a more advanced player, means that fitness evaluation is fundamentally noisy. Reliable fitness testing must use an appropriate number of games per opponent, as can be seen in the tournament grids of Figure 5.1, where 30 games per opponent is clearly unreliable in light of the results of using 200 games per opponent. In the round robin case, an individual's rank in the population is not determined until the end of all evaluation games. However, in the binary rank placement scheme it is important to have some clear idea of an individual's relative rank placement *after every individual*. This means that though there is a gain in dropping the number of opponents played from $O(N)$ to $O(\log(N))$, the necessary increase in games per opponent outweighs this.

For the single evaluator scheme, we see from Figure 6.5 that the best results obtained for a population size of 15, using 41 games per opponent, are almost as good as the round robin case with 11 games per opponent, indicating that optimisation can proceed well without a diverse set of opponents. However, by increasing the games played with single evaluator to just 81 we see a dramatic drop in learning speed, while the round robin scheme is learning well at 154 games per evaluation! In order to understand this it is important to note the sensitivity of the system to the k parameter, which dictates how often the teacher network will be replaced. This creates a new source for inefficiency, because the dynamic incremental evolution capability of coevolution is lost – something already proven to be necessary for coevolution in Section 4.2. Thus, when k is too low this causes teachers to be replaced too often, requiring constant re-evaluations and wasting many valuable computational cycles, while when k is too high we get a too-low plateau, a similar effect to the fixed evaluator evolution results in Figure 4.2. These effects can be seen in Figure 6.2, where 0.75 seems to be already too high, while 0.65 is too low. Because of this loss of coevolution's automatic incremental evolution, the single evaluator scheme is less efficient than simple round robin.

Despite their inability to outperform round robin, the binary rank placement and single evaluator results indicate that the important bottleneck to coevolution in the backgammon domain is not the number of opponents used in fitness evaluations, but the number of games. In other words, a diverse teaching set is not required, however it is very important to carefully deal with the noise inherent in the fitness tests.

Our final results indicate that we are able to get faster learning than the round robin scheme using the losers first strategy. This is due to successfully exploiting both the newcomers' rank distribution, which shows that 50% of the time our newcomer is unhelpful to evolution, and the domain transitivity, which means that if a newcomer is worse than the worst population member, it is worse than each of the other members of the population as well.

Darwen predicts that the number of games needed to evolve nonlinear structure using coevolution for backgammon players is infeasibly high [2]. In the final chapter we run longer experiments in the order of 20 million games each using the losers first strategy for both linear and nonlinear network structures, to see if losers first can evolve superior players in the nonlinear case within a feasible time-frame.

7 Nonlinear Optimisation

Finally, we use the losers first strategy for evolving nonlinear network structures to see if we are now able to outperform linear network structures.

We can achieve more efficient steady-state coevolution with losers first, however we need to choose appropriate population size and number of games per fitness evaluation. We have already seen that a large population is helpful (Section 4.1), and that as few as 41 games per evaluation can be used for successful coevolution of backgammon players (Section 6.4). Darwen uses round robin with a population size of 200, playing 10 games per opponent for a total of 1990 games per evaluation [2]. This many games per opponent is clearly excessive, in light of the 41 games per evaluation used by single evaluator. However, for longer testing it does appear desirable to use population sizes larger than just 15 as we have done so far. As can be seen in Figure 4.1, population size 141 starts learning more slowly than population size 15 but performs higher after 2 million games, achieving a better score and still improving at 8 million games.

In order to find an appropriate population size and games-per-match setting for longer nonlinear tests, we begin by contrasting Darwen's experimental setup – round robin experiments using 200 population size and 10 games per opponent, with the same using just 1 game per opponent. We then compare this with a population with 100 members and 1 game per opponent to see if 200 members is excessive, and finally we compare these to a losers first experiment also using 100 size population and 1 game per opponent.

With the results of this experiment we go on to run our final experiment, evolving nonlinear networks using 3 and 5 hidden nodes with the losers first coevolution strategy.

7.1 Experimental Setup

For the linear network structures we use the same structure as all previous experiments in this work. For nonlinear networks we introduce a hidden layer, testing one network with 3 and one with 5 hidden nodes. The initial comparison of larger populations compares 200 population size with 10 games per opponent, 200 population with 1 game per match, and 100 population with 1 game per match, using round robin. Additional runs using 100 population and 1 game per match were run using the losers first scheme, for reassurance that we still gain in efficiency at a larger population size.

The nonlinear experiments both use the losers first scheme, with 100 population size and 1 game per match. Because of this a slight modification was made to the losers first scheme for all experiments in this chapter. The initial match played against the current worst network in the population needs to be reliable. Playing just one game against the worst player and discontinuing all further evaluation on the result of that game is unwise due to the noise inherent in backgammon games. Thus, the first match against the worst player is always played using 11 games, while the remaining round robin games are played at the lower games-per-match setting.

The experiments using 3 hidden nodes were run for a total of 20 million games. However, experiments with 5 hidden nodes were so slow that they were abandoned after 5 million games. An extra round of experiments using losers first for linear structure was run to 20 million games for comparison, also using 100 population size

and 1 game per opponent.

All experiments in this section were run 10 times and averaged to get graphed results and confidence intervals.

7.2 Results

Results of the population size comparisons can be seen in Figure 7.1.

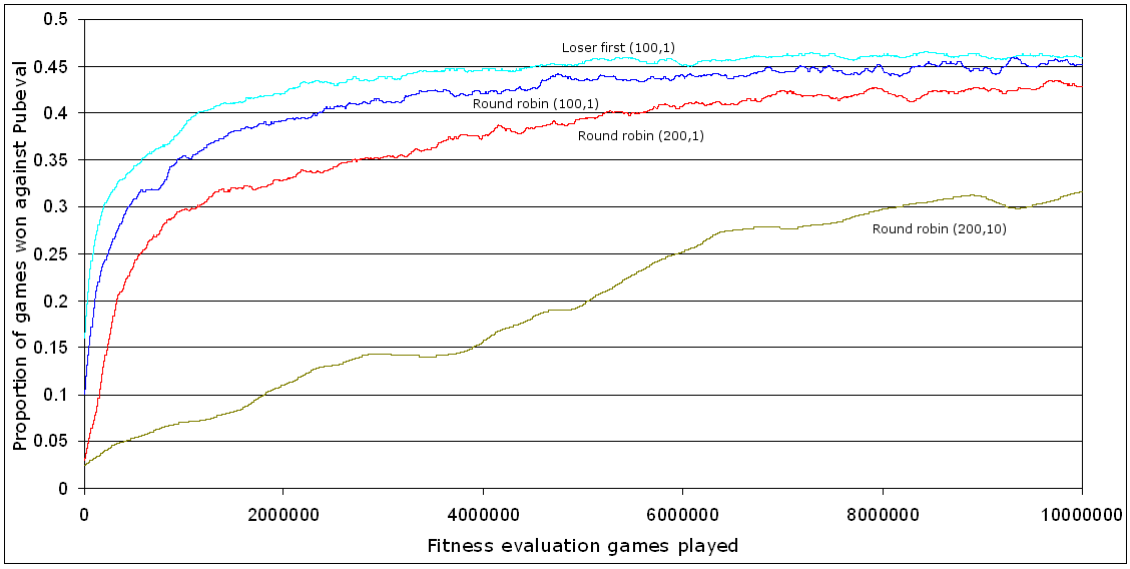


Figure 7.1: Comparison of population size 200 with 10 games per opponent to population 200 with 1 game per opponent, and population 100 with 1 game per opponent. Losers first with population 100 and 1 game per opponent is also compared. Confidence intervals for all curves at a given sample point are between 8 and 10% of the sample value.

We see that using 1 game per opponent instead of 10 for population size 200 is an order of magnitude faster at optimising. This confirms our prediction that 1990 games per evaluation is unnecessarily high. A population size of 100 with 1 game per opponent is even faster, using just 99 games per evaluation. Losers first still outperforms round robin at this population size.

Obviously further efficiency can be gained over the experimental settings used by Darwen by a more appropriate population size and games-per-match setting. Using population size 100 and 1 game per match, we test nonlinear structure for longer experiments as seen in Figure 7.2. We include results of a longer experiment using linear structure (0 hidden nodes) for comparison.

We wanted to run all the longer experiments for 20 million games each, however the heavy computational effort of EC became a problem. Networks with 3 hidden nodes have 3 times as many weights as a linear network structure, meaning 3 times more calculations for every move that is considered. Similarly, a network with 5 hidden nodes takes 5 times as many calculations per move considered. Because of this, though the linear network setup is able to play 30 million games after 120 hours, networks with 3 and 5 hidden nodes play just 10 million and 5.5 million respectively. Thus, after 120 hours[4] we decided to abandon the 5 hidden nodes network to focus on using 3 hidden nodes.

[4]120 hours is the maximum wall-time afforded us by the computing cluster used for these experi-

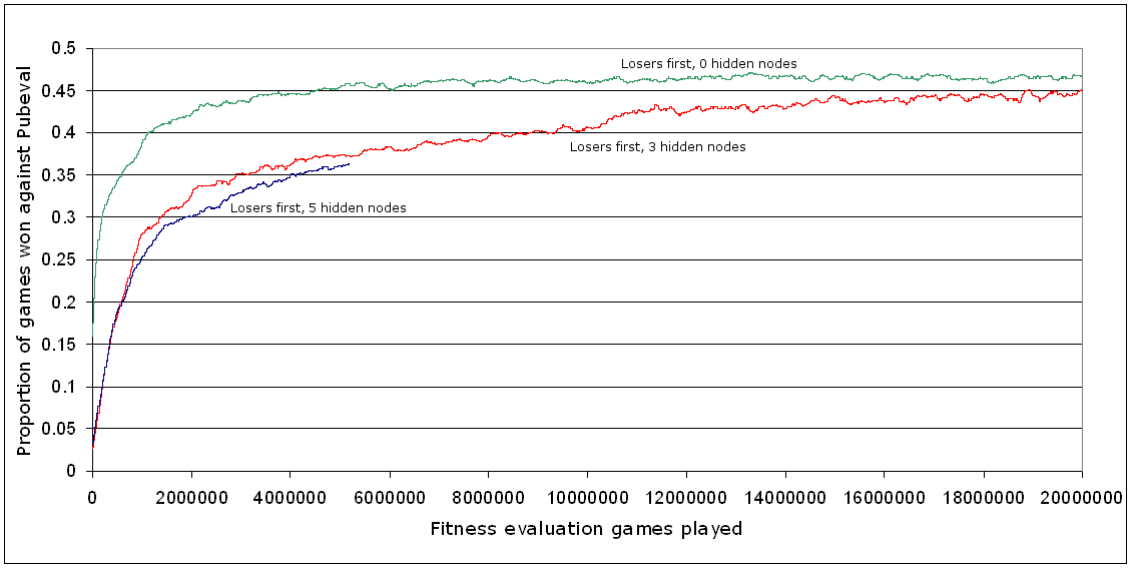


Figure 7.2: Comparison of losers first scheme using nonlinear networks with 3 and 5 hidden nodes, and a linear network with 0 hidden nodes. Experiments with 5 hidden nodes were abandoned due to infeasible computational times. Confidence intervals for all curves at a given sample point are between 8 and 10% of the mean value at the sampling point.

Figure 7.2 displays the results of the nonlinear experiments. Despite a clear efficiency gain using losers first, 100 population and just 1 game per match, we are still unable to evolve better backgammon players using nonlinear structure.

The network with 3 hidden nodes has not reached a clear plateau after 20 million games, and though it is still possible that after many more games these nonlinear structures could be optimised using coevolution to outperform linear networks at playing backgammon, the time necessary to test this hypothesis is unfortunately too long for the scope of this work, being in the order of one month per experiment.

Comparing to Tesauro's TD-Gammon [18] we see that his players used significantly more than 5 hidden nodes to achieve well above the linear networks skill level. With 10 hidden nodes TD-Gammon achieved a score of 52.7% against Pubeval, with 20 it achieved 57.1%, and with 40 it achieved 61.1% [19, Table 1]. Darwen demonstrates that coevolution can achieve higher than TD-learning in the backgammon domain for linear network structures, however to compete with Tesauro's nonlinear structures we would need to use 40 hidden nodes – clearly infeasible, as experiments using networks with just 5 hidden nodes are already running far too slowly. For nonlinear network structures it appears that evolutionary computation is simply too slow.

ments. In order to achieve 20 million games with 3 hidden nodes, the entire population was saved at the end of the first 120 hours, and the experiment was restarted using the restored population.

8 Discussion

This thesis presents an analysis of coevolution techniques in the backgammon domain, and demonstrates that it is possible to exploit transitive games for more efficient fitness evaluation.

Coevolution is useful for training backgammon players, benefiting from evolution's larger population sizes for a wider search than simpler optimisation strategies such as hillclimbing. Incremental evolution provided by coevolution is also very important, providing fitness tests that evolve in tandem with the players, encouraging far more growth than a fixed evaluation scheme.

However, the backgammon domain is almost purely transitive, and does not require a diverse teaching set to train from, which explains why no gain in optimisation is achieved through techniques such as fitness sharing or a hall of fame. Because of this, the single evaluator scheme can train backgammon players comparably as well as round robin coevolution.

The domain also suffers from a noisy fitness test, due to the nature of the game of backgammon, which means many games must be played to confidently gauge fitness values. This explains why binary rank placement fails to achieve more efficient optimisation than round robin – because it is important to have an accurate score against each opponent played, more games must be played per opponent and this offsets any benefit gained through playing less opponents.

The ranking distribution of coevolution in the backgammon domain can be exploited for more efficient optimisation. During coevolution, a newcomer will be worse than the worst player in the population approximately 50% of the time, meaning that it will not contribute useful genetic innovations to the evolving population and will not aid in optimisation. Other ranks are roughly evenly distributed. This has been successfully exploited for more efficient fitness testing using the losers first strategy. Losers first successfully evolves players more quickly and to a higher skill level than other fitness evaluation strategies, using linear network structures.

Finally, coevolution of nonlinear network structures still fails to achieve higher than linear network structures within a feasible time-frame, even using the losers first strategy. Darwen concludes that the number of games necessary to evolve nonlinear structure for the game of backgammon is impractically high [2]. By increasing the efficiency of coevolution using losers first, as well as a more optimal choice of parameters, we hoped to outperform our linear networks, however we remain unable to surpass Darwen's limitations.

8.1 Related Work

The research motivating this thesis comes primarily from three sources: Tesauro's TD-Gammon [18], Pollack & Blair's hillclimbing for backgammon training [10], and Darwen's coevolution of backgammon players [2].

Tesauro's TD-Gammon has already been compared to coevolution by Darwen in [2]. Though coevolution is able to outperform temporal difference learning for a linear network structure, albeit taking an order of magnitude more training games [2, 18], it remains unable to evolve nonlinear structure necessary to outperform linear networks.

Pollack & Blair achieved a surprising level of success training backgammon using hillclimbing techniques in [10]. The results of this thesis indicate that coevolution outperforms hillclimbing techniques through having a population size greater than 2. Pollack & Blair's networks plateau at an average score of 0.4, with a single network achieving 0.45, while the results of this work achieve a plateau as high as 0.48, with individuals scoring as high as 0.53. Pollack & Blair also develop intransitivities amongst later generational champions, which we show are due to noise in their fitness evaluations.

Finally, an increase in coevolution efficiency has been achieved over the round robin tournament method used by Darwen, using the losers first strategy. Also, Darwen's use of 1990 games per evaluation is unnecessarily high, by about a factor of 10. Using a population size of 100 and just 1 game per match, giving a total of 99 games per evaluation for normal round robin, provides further efficiency gains to coevolution. To illustrate this, Darwen's networks reach a score of 0.4 against Pubeval after 8 million games [2, Figure 4], while the losers first strategy reaches 0.4 in just over 1 million games (Figure 7.2). Despite these gains in efficiency, Darwen's observation that nonlinear structure is not being learnt through coevolution still holds.

8.2 Directions for Future Research

This work shows that the backgammon domain is noisy yet transitive, and that the ranking distribution of round robin coevolution can be effectively exploited for more efficient fitness evaluation, however much more by way of analysis and exploitation can be done in this domain.

Techniques useful for noisy domains have yet to be applied to these strategies, such as Darwen's and Pollack & Blair's use of "canned dice", whereby pairs of games are played using the same pseudo-random number sequence for dice rolls, with players taking turns playing the opening move [10]. Such techniques could go far in decreasing the number of games necessary for learning backgammon.

Of course, optimality of software and experimental setup also remains an issue. Our experiments ran for no longer than 20 million games each, which may not be long enough for rigorous testing of a coevolutionary method, yet such experiments typically needed an entire week to run. Methods for distributing evolutionary calculations over parallel processors, as well as optimisation of code for backgammon games as well as network calculations, could go far in assisting further analysis.

Further work in this domain could also involve a more intelligent approach to learning nonlinear structure for neural networks. Topological evolution, which was briefly experimented with in the early stages of this thesis, remains to be properly investigated, as well as directed search methods for slowly increasing numbers of hidden nodes. Combinations of topological evolution with reinforcement learning techniques such as TD-Gammon's temporal difference learning method might also yield interesting results, given the success of TD-Gammon.

Finally, the analyses and new approaches to fitness evaluation in this work are certainly not limited to the game of backgammon. Clearly, careful domain analysis offers possibilities for more efficient coevolution in any domain. While new research often focuses on the development and proof of superior algorithms for generalised cases,

this work demonstrates the importance of more rigorous domain analyses to aid in our understanding of why some techniques work, why some do not, and how we can best tailor algorithms to work in specific domains.

Appendix: Algorithm Parameters

For reproducibility, we present the evolutionary algorithm parameters used in this work here.

The algorithm used was based on the NEAT algorithm [13], with topological mutation rates set to zero. Parameter settings used are as follows:

- All parent species and organism selection done by roulette wheel selection.
- Unlimited number of species with a similarity threshold of 0.5.
- Probability that a new network is obtained by mating set to 0.7. If mating is not chosen, mutation is automatically performed.
- Probability that mating takes place between 2 parents from different species set to 0.25.
- Probability of mutation occurring after mating set to 0.4.
- Probability of single point crossover (rather than multi-point) set to 0.8.
- Probability of mutating link weights set to 0.9.
  - 0.75 chance of a link mutation incrementing link weight by $\delta_i \in (-0.5, 0.5)$. Otherwise, link mutation replaces link weight by $\delta_r \in (-1, 1)$. δ_i and δ_r chosen from even distributions.
- All other mutation probabilities set to 0.

References

[1] Berliner, H. (1977), Experiences in evaluation with BKG: A program that plays backgammon, in *Proceedings HCAI-77*, Cambridge, MA, pp 428-433.

[2] Darwen, Paul J. (2001), Why Coevolution beats Temporal Difference learning at Backgammon for a linear architecture, but not a nonlinear architecture, in *Proceedings of the 2001 Congress on Evolutionary Computation*.

[3] de Jong & Pollack (2004), Ideal Evaluation from Coevolution, in *Evolutionary Computation* Vol 12, Number 2.

[4] Ficici & Pollack (2003), A Game-theoretic Memory Mechanism for Coevolution. in *Proceedings of the 2003 Genetic and Evolutionary Computation Conference*, pp 286-297.

[5] Magriel, Paul (1976), *Backgammon*. Quadrangle/The New York Times Book Co. ISBN 0-8129-0615-2.

[6] Minsky & Papert (1969), Perceptrons, Cambridge, MA: MIT Press.

[7] Mitchell, Tom M. (1997), *Machine Learning*, McGraw-Hill, ISBN 0-070-42807-7.

[8] Monroy, Stanley, Miikkulainen (2006), Coevolution of Neural Networks using a Layered Pareto Archive, in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pp 329-336.

[9] Morehead, Morehead, & Mott-smith, *Hoyle's Rules of Games* (2001), Third Revised and Updated Edition, Signet, 321-330. ISBN 0-451-20484-0.

[10] Pollack & Blair (1998), Coevolution in the Successful Learning of Backgammon Strategy, in *Machine Learning* Vol 32, Number 3, pp 225-240.

[11] Rosin & Belew (1997), New Methods for Competitive Coevolution, in *Evolutionary Computation* Vol 5, Number 1, pp 1-29.

[12] Russell, Stuart J. & Norvig, Peter (2003), *Artificial Intelligence: A Modern Approach* (2nd ed.), Upper Saddle River, NJ: Prentice Hall, ISBN 0-137-90395-2.

[13] Stanley & Miikkulainen (2002), Evolving Neural Networks through Augmenting Topologies, MIT Press Journals, in *Evolutionary Computation* Vol 10, Number 2, pp 99-127.

[14] Sutton & Barto (1998), *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, ISBN 0-262-19398-1.
Also online: `http://www.cs.ualberta.ca/~sutton/book/ebook/the-book.html`

[15] Tesauro, Sejnowski (1989), A Parallel Network that Learns Backgammon, in *Artificial Intelligence* Vol 39, Issue 3, pp 357 - 390.

[16] Tesauro (1989), Neurogammon Wins Computer Olympiad, in *Neural Computation* Vol 1, pp 321-323.

[17] Tesauro (1993), FTPable benchmark evaluation function, Forum archives of rec.games.backgammon, retrieved 21-07-2008.
`http://www.bkgm.com/rgb/rgb.cgi?view+610`

[18] Tesauro, G. (1995), Temporal Difference Learning in TD-Gammon, in *Communications of the ACM*, March 1995 / Vol. 38, Number 3.

[19] Tesauro, G. (1998), Comments on "Coevolution in the Successful Learning of Backgammon Strategy", in *Machine Learning* Vol 32, Number 3, pp241-243.

[20] Winkeler & Manjunath (1998), Incremental Evolution in Genetic Programming, in *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pp 403-411.