A Coevolutionary Model for The Virus Game

P.I.Cowling, M.H.Naveed and M.A. Hossain
MOSAIC Research Centre, Department of Computing,
University of Bradford
Bradford, BD7 1DP, UK.
E-mail: P.I.Cowling, M.H.Naveed and M.A.Hossain1@bradford.ac.uk

***Abstract*— In this paper, coevolution is used to evolve Artificial Neural Networks (ANN) which evaluate board positions of a two player zero-sum game (The Virus Game). The coevolved neural networks play at a level that beats a group of strong hand-crafted AI players. We investigate the performance of coevolution starting from random initial weights and starting with weights that are tuned by gradient based adaptive learning methods (Backpropagation, RPROP and iRPROP). The results of coevolutionary experiments show that pre training of the population is highly effective in this case.**

I. INTRODUCTION

In biology, coevolution is the mutual evolutionary influence between two species that become dependent on each other. Each species in a coevolutionary relationship exerts selective pressures on the other species. Coevolution occurs if the traits of one species *A* have evolved due to the presence of a second species *B* and vice versa. This natural phenomenon has motivated AI researchers to apply coevolution in solving different types of problems where two or more entities are interacting with each other. Coevolution is an unsupervised learning method that requires only relative measurement of phenotype performance, well-suited to the game-playing domain.

The gradient-based learning methods: Backpropagation [17], resilient backpropagation (RPROP) [16] and improved resilient backpropagation (iRPROP) [9] are supervised learning methods. They use a delta rule and can be applied to the problem of learning neural network weights to give a network which produces the desired outputs with minimised error.

Games continue to be important domains for investigating problem solving techniques [13]. Games offer tremendous complexity in a computer manageable form and need sophisticated AI methods to play at expert level. Board games like Chess [4], checkers [8], Othello [24] and backgammon [22] have been used to explore new ideas in AI. We will survey this work late in this section. In this paper, we used the "Virus Game" as a testbed to explore coevolutionary ideas.

The "Virus Game" [6] [7] [10] is a two-person perfect information board game of skill. The game is played on a square board. The start position of the game is shown in Figure 1. The player who always starts the game is the *Black Player* and the other player is the *White Player*.

In the Virus Game, there are two kinds of moves available for each turn. The first kind of move is *grow move* or *one step move*. In this kind of move, a player moves a piece of his colour to an empty position adjacent to its current position. The positions are adjacent if their borders or corners are adjacent. The result of this move reproduces the moving piece and occupies both positions, the new position (which was empty) and the old position. The grow move is shown in Figure 2. The second kind of move is called *jump move* or *two step-move*. In this case, a player moves a piece to an empty position which is two squares away from its current position via an empty square. The piece leaves the old position empty and occupies the new position. Figure 3 shows a jump move. In either case, all opposing pieces adjacent to the new player's piece change colour. Players alternate, moving only one piece per turn. The game ends when neither player can move. The player with the greatest number of pieces is the winner. The game is declared a draw if both players have the same number of pieces at game end.

The Virus Game has higher branching factor than chess, draughts and Othello and appears to be a difficult game for a human player to play well despite its simple rules [7]. We have a strong pool of hand-crafted AI players for the Virus Game, each written by a different person since these were used in an AI-writing competition [6].Competitive learning was initially explored by Samuel [20] to adjust the parameters of a deterministic evaluation function in a checkers playing computer program. Tesauro [22] used the temporal-difference learning approach to evolve a backgammon-playing neural network. Tesauro's TD-Gammon yields a computer playing backgammon program of world-champion strength. Coevolutionary competitive learning is explored for the Repeated Prisoner's Dilemma (RPD) by Axelrod [2] and Miller [11]. Axelrod evolve RPD playing strategies using a fixed environment (i.e. using eight fixed opponents) while Miller coevolved the RPD strategies by playing each strategy against every other strategy and itself in a population. According to the results shown by

Miller, the best evolved RPD playing strategies in his work performed well against strong strategies (like Tit-for-Tat) taken from Axelrod's work. Axelrod and Miller used Genetic Algorithms to evolve RPD playing strategies.

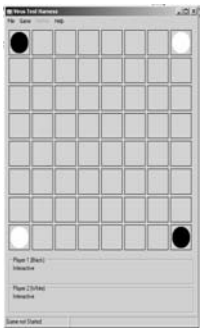

Fig. 1. The starting position of the Virus Game Board

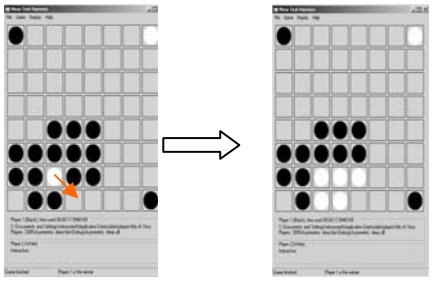

Fig. 2. This figure represents a one-step or grow move. The White player is to move the white piece at position (3, 2) moves to position (4, 1). This move gives another white piece at (4, 1) and captures the pieces of opposite colour at positions adjacent to (4, 1).

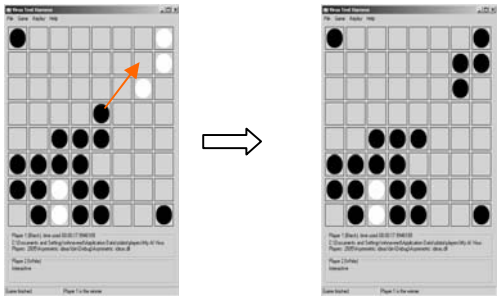

Fig. 3. This figure shows a jump move. Black is to move and chooses to move the piece at (5, 5) to (7, 7) which captures white pieces in the squares surrounding square (7, 7).

Angeline and Pollack [1] used competitive coevolution with Tic Tac Toe as a testbed. They introduced a competitive fitness function where the total number of competitions in a generation is n-1 for a population of size n. This gives a small number of competitions as compared to the competitions in Miller's work [11] where the total number of competitions per generation is n^2. This competitive fitness function saves a considerable amount of CPU time. Smith and Gray [19] introduced a competitive function where there are n/2 competitions per generation for a population of size n. Smith and Gray applied coevolution to Othello. The weights of a deterministic evaluation function are evolved using a co-adapted GA with explicit fitness sharing. The coevolved evaluation function may not be very strong but the approach is notable for the formation of stable niches (i.e. stable groups) during the evolutionary process. The individuals of each group in a generation have similar characteristics and the results show that the individuals in a group continuously evolve during the coevolutionary process.

Potter and De Jong [14] explored cooperative coevolution for function optimisation. They introduced Cooperative Coevolutionary Genetic Algorithms (CCGAs) where a group of subpopulations is maintained which interact with each other in a modular fashion. Each subpopulation represents a partial solution and combining the members of all subpopulations gives complete solutions. The number of subpopulations is not fixed in CCGAs and there is no migration between the subpopulations. The competition for evolution in CCGAs exists among the members of each subpopulation and each subpopulation has its own evolutionary algorithm. The results show that CCGAs have better performance than standard GAs. This approach is also notable for its natural mapping onto a client server architecture where subpopulations can be coevolved on different network machines in parallel (simultaneously) and each subpopulation can have its own evolutionary algorithm.

The effectiveness of CCGAs in solving complex problems is explored by Potter et al [15] to evolve the sequential decision rules which control the behavior of a simulated robot. In this case, the empirical results demonstrate that cooperative coevolution has better rule learning speed than non-coevolutionary systems and it promotes the formation of stable niches which provide evidence of cooperation among subpopulations.

Anaconda [8] is a checkers playing neural network which is evolved using competitive coevolution.The authors have used Evolutionary Programming to coevolve the neural networks in a competitive environment and the strongest neural network, Anaconda, is rated at expert level according to a tournament conducted at website www.zone.com.

This paper investigates the effectiveness of two coevolutionary approaches. In the first approach, initial populations, of varying sizes, containing random neural networks, are evolved against 10 strong hand-crafted AI players. The weights of neural networks are evolved using a Genetic Algorithm. In the second coevolutionary approach, a large number of neural networks are trained using gradient based learning methods under the supervision of 10 fixed hand-crafted AI players. The trained neural networks are then coevolved against the same and different fixed opponents. Thus we are able to investigate whether the combination of coevolution and supervised learning is effective.

The paper has the following structure. Section II describes our experimental design. Section III contains results and their analysis. Section IV concludes the paper.

II. EXPERIMENTAL DESIGN

A. Design of the neural network

The design of a neural network architecture is a complex task. The selection of an appropriate architecture for a given problem is very important because the learning capabilities of a neural network depend heavily on its architecture [25]. In our work, we used different neural network architectures in our experiments. The selection of the final architecture is made on the basis of its performance in these initial experiments. The architecture of our neural networks is represented by I-H_1-H_2-O where 'I' represents number of input units, 'H_1' means the number of hidden neurons in first hidden layer, 'H_2' represents total number of hidden neurons in second hidden layer and 'O' represents number of output neurons. The architectures that were initially explored are 64-20-10-1, 64-25-10-1, 64-30-15-1, 64-35-15-1, 64-44-20-1, 64-50-25-1, 64-58-27-1, 64-66-34-1 and 64-71-42-1. The small neural networks show poor performance but use low CPU time while large neural networks have good performance with high CPU time. The performance for each architecture was evaluated by coevolving a population of 20 neural networks (with random weights) against 10 fixed AI players. Three runs for each architecture were performed to allow for stochastic variations. The average value of maximum score in 500 generations represents the fitness of the architecture. The average value for architecture 64-58-27-1 is higher than the average values of other architectures. The first five architectures mentioned earlier in this subsection could not converge to a solution within 500 generations (for all three runs) where they can beat all opponents. The neural networks with last three architectures defeated all opponents within 10-16 generations of coevolution. The architecture 64-58-27-1 (smallest among last three architectures) is used in all subsequent experiments.

In the selected neural networks architecture, there are 64 input units where each input unit is associated with a square of board. A location-based input encoding scheme [26] is used where each input unit is associated with a square of board. If the corresponding square of an input unit has a black piece on the board then its value is assigned as '1', if square is occupied with white piece the input value for this unit is '-1' and value '0' is used for empty square.

All hidden neurons have sigmoid [17] activation function. All weights are encoded by real numbers in the range [-0.5, 0.5]. The output neuron has linear activation function and gives a real number as the output of the neural network.

B. Evolutionary Method

A Genetic Algorithm (GA) is used to evolve the connection weights of a population of neural networks. A chromosome contains all connection weights of one neural network as shown in Figure 4.

W_1	W_2	…		W_{5363}

Fig. 4. A representation of a chromosome where W_1, W_2, … W_{5363} are real-valued connection weights.

The fitness of a chromosome is the total score of its corresponding neural network in playing two games, one as a black and other as white, with each of 10 fixed AI opponents. If a neural network wins a game against an opponent, its score is increased by 3; if the match is draw then score is increased by 1 and for loss there is no change in the score. Each neural network plays twenty games per generation and its total score after twenty games is used to measure its fitness. All games are run to 1-ply only to keep search times manageable.

The Selection operator applies a rank selection method [12]. At the end of the tournament for a given generation, all chromosomes are arranged in a descending order according to fitness score. From the sorted list of N chromosomes, the top N/3 chromosomes are selected for sexual reproduction. These selected chromosomes are paired to reproduce 2N/3 new chromosomes through a two-point crossover operation. The process of two-point crossover is shown in Figure 5 where each chromosome is treated as a circular entity. The Gaussian Mutation Operator [23] is applied to each new offspring. The weights of a newly created neural network are changed using following equations.

$$\begin{aligned} \boldsymbol{W_i^m} &= \boldsymbol{W_i} + \boldsymbol{N}(0, \sigma_i^m), \\ \sigma_i^m &= \sigma_i \times \exp(\boldsymbol{N}(0, (\sqrt{\boldsymbol{L}})^{-1}), \quad \boldsymbol{i} = 1,2,..\boldsymbol{L} \end{aligned} \tag{1}$$

Where $\boldsymbol{W_i}$ is the i[th] weight of a given chromosome before mutation operation and $\boldsymbol{W_i^m}$ is the i[th] weight of that chromosome after mutation operation. The term σ_i is self-adaptive parameter vector related to weight $\boldsymbol{W_i}$ of a chromosome. Initially it is set to 0.10 for all *i*. The selection of this value of mutation parameter is based upon [8] where they used 0.05 for σ_i and initial connection weights between [-0.2, 0.2]. The total length of a chromosome is *L*. In [18], σ is initialised to the value $1/\sqrt{L}$. N(0, σ) denotes a Gaussian random variable with mean 0 and standard deviation of σ.

The sets of parameters which are examined in our coevolutionary model are shown in Table 1.

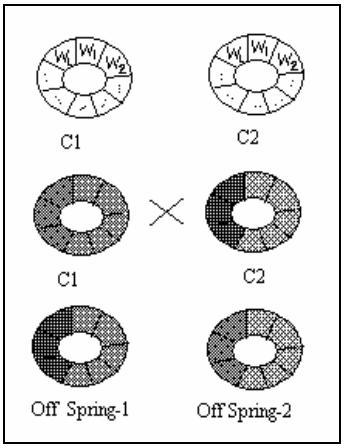

Fig. 5. The process of two-point crossover. C1 and C2 are the chromosomes selected for sexual reproduction.

TABLE.1: PARAMETERS FOR GA EXPERIMENTS

POPULATION SIZE	CROSS OVER RATE	MUTATION SELF-ADAPTIVE PARAMETER
6	0%	0.10
6	20%	0.10
15	0%	0.10
15	20%	0.10
20	0%	0.10
20	20%	0.10
30	0%	0.10
30	20%	0.10
50	0%	0.10
50	20%	0.10

C. Supervised learning Methods

Neural networks are trained under the supervision of 10 hand-crafted AI players using gradient-based techniques: Backpropagation (BP), RPROP and iRPROP. In this case, 10 different training sets are created where each training set contains a large number of board positions and their evaluation values as determined by a hand-crafted AI player. Therefore, each neural network is trained to learn the evaluation function of a given hand-crafted AI player. Each neural network is trained using one training set. Four neural networks are trained using iRPROP, three are trained using RPROP and three are trained using BP, for each hand-crafted AI player. BP uses a learning rate of 0.01. The learning parameters for RPROP and iRPROP are set to η^+=1.2, η^-=0.50, Δ_o=0.5, Δ_{min}=0 and Δ_{max}=50. The selection of the learning parameters of gradient-based learning methods is made according to the recommended values by authors in [21], [16] and [9]. Thus a pool of 100 neural networks is trained under the supervision of 10 different training data sets (each based on a different hand-crafted AI player).

D. Coevolutionary Model

Our coevolutionary model uses two different approaches for generating the starting population. In the first approach, an initial population of neural networks with random weights is coevolved against ten fixed opponents. The coevolution of neural networks continues until at least one neural network in a given population beats all of the fixed opponents or there is no improvement in the scores of the best neural network for 10 consecutive generations. All games in this coevolutionary approach are played at 1-ply by both neural networks and fixed opponents.

In the second approach an initial population of trained neural networks is used. These neural networks are selected from the pool of 100 pre-trained neural networks. If the population size is less than or equal to 10, all neural networks used are based on different hand-crafted AI players. If population size is greater than 10, these are selected randomly in such a way as to ensure that a neural network trained by each of the 10 hand-crafted AI players is in the starting population. Therefore two populations of size 20 would have 10 identical and 10 different neural networks in the initial population. The trained neural networks are coevolved against 10 fixed opponents. The stopping condition and ply depth is same as used in first approach. So, the difference between two approaches is that the second approach combines two machine learning techniques i.e. gradient-based learning and coevolution.

To assess the generality of approach, we also test against another 10 hand-crafted AI players which are not used in pre-training or coevolution.

III. RESULTS AND ANALYSIS

The results obtained from the randomly initialised neural networks are summarized in table 2 while the performance of the best of these coevolved neural networks against 10 hand-crafted AI players is shown in table 3. All experiments are run on Pentium IV 1.2 GHz using C#.Net running under Windows XP.

TABLE. 2. RESULTS OF COEVOLUTION WITH AN INITIAL POPULATION OF RANDOMLY CREATED NEURAL NETWORKS. COLUMNS 3, 4 AND 5 SHOW PERFORMANCE AGAINST TEN HAND-CRAFTED OPPONENTS AVERAGED OVER 10 COEVOLUTIONARY RUNS

POP SIZE	CROSS OVER RATE	MEAN OF MAX	MEAN OF MIN	MEAN OF MEAN	MEAN OF GENERATION	MEAN CPU TIME (SEC)
6	0%	3	0	2	11	9
6	20%	18	0	9	10	13
15	0%	9	0	1	10	28
15	20%	6	0	1	11	40
20	0%	12	0	5	11	53
20	20%	30	0	3	6	57
30	0%	30	0	16	11	61
30	20%	60	0	34	6	72
50	0%	21	0	22	11	132
50	20%	60	0	35	5	72

Table 2 shows the results of coevolution with populations starting from randomly created neural networks. The column *"Mean of Generation"* represents average number of generations over 10 runs and "mean CPU time" represents

usage of average CPU time in seconds by a population during coevolutionary process over 10 runs.

We can see that small populations give much worse results than large ones. The difference between the performance of small and large populations is probably due to the number of parallel directions for exploring potential solutions in the search space. The experimental results also demonstrate that small populations use less CPU time than large populations, essentially since less neural network evolution are required..

The average maximum scores with 0% and 20% crossover rates demonstrate that neural networks evolved using crossover generally have higher scores than those which do not. The crossover operation appears to help the evolutionary process to explore a more interesting region of the search space. The crossover operator speeds convergence, and the population converges to better solution than without crossover. The mean of min and mean values show that diverse populations are maintained.

TABLE. 3. THE SCORE OF THE STRONGEST NEURAL NETWORK FROM EACH COEVOLVED POPULATION STARTING FROM AN INITIAL POPULATION OF RANDOMLY CREATED NEURAL NETWORKS AGAINST 10 HAND-CRAFTED AI OPPONENTS.

POP SIZE	CROSS OVER RATE	PLAYING AS BLACK		PLAYING AS WHITE	
		WIN	DRAW	WIN	DRAW
6	0%	10	0	0	0
6	20%	10	0	10	0
15	0%	10	0	0	0
15	20%	10	0	0	0
20	0%	10	0	0	0
20	20%	10	0	10	0
30	0%	10	0	0	10
30	20%	10	0	10	0
50	0%	10	0	0	0
50	20%	10	0	10	0

Table 3 summarises the results of the best evolved neural network from each population where coevolution was started from an initial population of randomly generated neural networks. Each evolved neural network plays as black and white against ten hand-crafted AI, as in table 2. All the best neural networks with 0% crossover rate have won games as black against all AI opponents but none is able to win when playing as white. The best-evolved neural networks of all populations (except population of size 15) with 20% crossover rate won all games when playing as black and white. Table 2 and 3 provide clear evidence for the effectiveness of the crossover operator, and support the advantage of Black and White in the Virus game, at least for approaches using 1-ply search.

Table 4 shows the mean of maximum, mean, and minimum scores of coevolved neural networks with a starting population of pre-trained neural networks. The table also shows the mean number of generations and CPU time (in seconds) for each population with different crossover rates. Again we see that small populations perform more poorly than larger populations. If we compare the results of tables 2 and 4, it can be construed that starting from a population of pre trained neural networks gives better and more consistent performance (in terms of playing strength) than starting from a population of random neural networks. Large sized populations (started with initial population of pre trained neural networks) use a smaller number of generations but still require more CPU time during the coevolution of neural networks than small size populations. The values of mean of minimum and maximum scores show the diversity is maintained in the population. Note that CPU times in this case include the time need for learning of the initial population

Generally, we see from table 4 that coevolution with crossover needs fewer generations and less CPU time than without crossover.

TABLE. 4. RESULTS OF COEVOLUTION WITH AN INITIAL POPULATION OF PRE-TRAINED NEURAL NETWORKS. COLUMNS 3, 4 AND 5 SHOW PERFORMANCE AGAINST TEN HAND-CRAFTED OPPONENTS AVERAGED OVER 10 COEVOLUTIONARY RUNS.

POP SIZE	CROSS OVER RATE	MEAN OF MAX	MEAN OF MIN	MEAN OF MEAN	MEAN OF GENERATIONS	MEAN CPU TIME (SEC)
6	0%	40	0	28	13	12
6	20%	60	0	31	5	7
15	0%	60	0	22	3	18
15	20%	60	0	28	1	18
20	0%	60	0	26	3	31
20	20%	60	0	31	1	25
30	0%	60	0	29	2	66
30	20%	60	0	36	1	55
50	0%	60	0	32	1	84
50	20%	60	0	34	1	84

Table 5 gives the mean results of the maximum, mean and minimum scores of the initial populations before the start of evolution. The results after the coevolution are shown in table 4. In many cases evolution is not required, in spite of the fact that the performance of the AI players is variable, as table 7 shows.

TABLE 5. RESULTS FOR PRE TRAINED NEURAL NETWORKS PRIOR TO EVOLUTION.

POP SIZE	CROSS OVER RATE	MEAN OF MAX	MEAN OF MIN	MEAN OF MEAN
6	0%	40	0	30
6	20%	40	0	30
15	0%	55	0	40
15	20%	55	0	44
20	0%	55	0	40
20	20%	60	0	45
30	0%	55	0	44
30	20%	60	0	45
50	0%	60	10	35
50	20%	60	0	40

The results of table 5 show that the populations of large size (starting from the initial population of pre trained neural

networks) generally have at least one pre trained neural network which beats all the opponents before the start of coevolution when playing as black and white. We see from table 4 that the values of mean of mean scores are reduced during coevolutionary process when compared with the values in table 5, and that average mean scores with crossover are larger than average mean scores without crossover.

In order to further investigate the generality of populations starting from an initial population of pre trained neural networks, we introduced 10 more hand-crafted AI opponents which were not used in the training of the initial population. Table 6 gives the average results of coevolution with different populations starting from initial population of trained neural networks where neural networks play as black and white against 20 AI opponents in each generation which include the 10 previously unseen opponents. In this case, pre trained neural networks in larger populations were able to beat not only AI players used for initialisation but also the AI players which were unseen by them during the supervised training of initial population of pre trained neural networks. The results shown in tables 6 and 4 are quite similar. The results of these both tables with populations of small sizes show that coevolution with 20% crossover rate has better performance than coevolution with 0% crossover rate. The coevolution of populations of large size has same performance with both crossover rates (essentially since the initial populations were very strong).

TABLE . 6. SUMMARY OF RESULTS FROM 10 RUNS OF COEVOLUTION OF PRE-TRAINED NEURAL NETWORKS USING 20 OPPONENTS.

POP SIZE	CROSS OVER RATE	MEAN OF MAX	MEAN OF MIN	MEAN OF MEAN	MEAN OF GENER ATION	MEAN CPU TIME (SEC)
6	0%	80	0	66	12	102
6	20%	120	60	73	4	96
15	0%	120	0	45	2	48
15	20%	120	0	52	1	48
20	0%	120	0	45	2	48
20	20%	120	0	55	1	78
30	0%	120	0	59	1	102
30	20%	120	0	70	1	132
50	0%	120	0	87	1	168
50	20%	120	0	95	1	210

Table 7 shows the results of tournament among the 10 AI players used for population initialisation. The results of the tournament reveal that all these players have similar strength with no obvious champion. According to the table 5, in the populations of large size, there is at least one trained neural network, which beats all 10 opponents when playing as black and white against them. According to Tesauro and Sejnowski [26], a trained neural network can play as well as the teacher that trained it. Our results appear to indicate that a trained network can greatly outperform its teacher in this case.

TABLE. 7. SUMMARY OF RESULTS OF TOURNAMENT AMONG FIXED AI PLAYERS. COLUMNS 2 AND 3 SHOW THE PERFORMANCE OF EACH HAND-CRAFTED AI PLAYER AGAINST 9 OTHER HAND-CRAFTED AI PLAYERS.

FIXED PLAYER NO	PLAYING AS BLACK		PLAYING AS WHITE	
	WIN	DRAW	WIN	DRAW
1	4	1	4	1
2	1	0	7	0
3	7	1	5	0
4	3	0	6	0
5	5	0	4	0
6	6	0	5	0
7	1	0	7	1
8	***7***	***1***	***5***	***1***
9	3	0	7	0
10	6	1	6	0

TABLE. 8. SUMMARY OF TOURNAMENT RESULTS FOR COEVOLVED NEURAL NETWORKS FROM INITIAL POPULATIONS OF RANDOMLY CREATED NEURAL NETWORKS. COLUMNS 2 AND 3 SHOW THE PERFORMANCE OF EACH ANN AGAINST EACH OTHER ANN.

PLAYERS NO	NO OF GAMES WON AS BLACK PLAYER	NO OF GAMES WON AS WHITE PLAYER	TOTAL SCORE	RANK (1-20)
$R_{6,0}$	4	4	26	18
$R_{6,20}$	1	7	32	16
$R_{15,0}$	4	5	28	17
$R_{15,20}$	5	2	22	19
$R_{20,0}$	5	5	32	15
$R_{20,20}$	9	6	47	12
$R_{30,0}$	9	9	54	11
$R_{30,20}$	2	4	20	20
$R_{50,0}$	7	7	42	13
$R_{50,20}$	7	6	39	14

TABLE. 9. SUMMARY OF TOURNAMENT RESULTS FOR COEVOLVED NEURAL NETWORKS FROM POPULATIONS STARTING FROM PRE TRAINED NEURAL NETWORKS. COLUMNS 2 AND 3 SHOW THE PERFORMANCE OF EACH ANN AGAINST EACH OTHER ANN

PLAYERS NO	NO OF GAMES WON AS BLACK PLAYER	NO OF GAMES WON AS WHITE PLAYER	TOTAL SCORE	RANK (1-20)
$P_{6,0}$	10	10	63	10
$P_{6,20}$	14	13	83	5
$P_{15,0}$	16	15	93	1
$P_{15,20}$	16	15	93	2
$P_{20,0}$	16	15	93	3
$P_{20,20}$	16	7	70	8
$P_{30,0}$	16	7	70	9
$P_{30,20}$	13	13	84	4
$P_{50,0}$	16	8	73	6
$P_{50,20}$	16	8	73	7

Tables 8 and 9 present the results for a tournament between the best player from each set of experiments starting with random initial weights, R6,0, R6,20, ..., R50,20, and the best player from each set of experiments starting with pre-trained networks, P6,0, P6,20, ..., P50,20. Each of these players plays 19 games as black against each other player,

and 19 games as white against each other player. Since every one of P6,0, P6,20, ..., P50,20 is ranked higher than every one of R6,0, R6,20, ..., R50,20 this provides very clear evidence for the superiority of pre-training, particularly since the training times for pre-trained networks were significantly smaller than for the random initial population. Within R6,0, R6,20, ..., R50,20 and within P6,0, P6,20, ..., P50,20 we see significant variations in performance. It is clear that there are still significant variations in the phenotype among these best players.

IV. CONCLUSION

A coevolutionary model is presented where artificial neural networks adapt and learn to play the Virus Game using competitions with fixed strong hand-crafted (deterministic) AI players. The neural networks are provided with only raw board positions.

The results presented in this paper evidence the potential advantages of a combination of coevolution and supervised learning techniques for building knowledge into artificial neural networks. The combination of coevolution and gradient-based learning techniques gives improved playing performance and faster learning when compared to either approach combined in isolation. It would be interesting to explore the combination of gradient-based learning techniques and coevolution for other games and in the case where a deeper search is used.

The Genetic Operators of Crossover and Mutation are also analysed and we show that an evolutionary algorithm with crossover has much better performance than one with mutation alone in most experiments. In future we aim to investigate the dynamics which make crossover an effective operator.

REFERENCES

[1] P.J. Angeline and J.B. Pollack, "Competitive Environments Evolve Better Solutions for Complex Tasks", in the proceedings of 5th International Conference on Genetic Algorithms (GAs-93), pp. 264-270, 1993.

[2] R. Axelrod, "The Evolution of Strategies in the Iterated Prisoner's Dilemma", Genetic Algorithms and Simulated Annealing, in Lawrence Davis (ed.), Morgan Kaufmann, pp. 32-41, 1987.

[3] D. Beasley, D.R. Bull and R.R. Martin, "An Overview of Genetic Algorithms: Part2, Research Topics", University Computing, Vol. 15, No. 4, pp. 170-181, 1993.

[4] M. Campbell, A.J. Haone Jr., F-h. Hsu, "Deep Blue", Artificial Intelligence, Vol.134, pp.57-83, 2002.

[5] S.Y. Chong, M.K. Tan and J.D. White, "Observing the Evolution of Neural Networks Learning to Play the Game of Othello", IEEE Transactions on Evolutionary Computation, Vol.9, No.3, pp. 240-251, 2005.

[6] P.I. Cowling, R. Fennell , R. Hogg , G. King, P. Rhodes, N. Sephton, "Using Bugs and Viruses to Teach Artificial Intelligence", in the proceedings of 5th Game-on International Conference on Computer Games: Artificial Intelligence, Design and Education, pp. 360-364, 2004.

[7] P.I. Cowling, "Board Evaluation for the Virus Game", in the proceeding of IEEE 2005 Symposium on computational Intelligence and Games (CIG'05), Graham Kendall and Simon Lucas (editors), pp. 59-65, 2005.

[8] D.B. Fogel and K. Chellapilla, "Verifying Anaconda's expert rating by competing against Chinook: experiments in co-evolving a neural checkers player", Neurocomputing, Vol.42, pp.69-86, 2002.

[9] C. Igel and M. Husken, "Empirical Evaluation of the Improved RPROP Learning Algorithms", Neurocomputing, Vol. 50C, pp.105-123, 2003.

[10] J. Matthews, "Virus Game Project", http://www.generation5.org/content/2000/virus.asp, 2000.

[11] J.H. Miller, "The Coevolution of automata in the repeated prisoner's dilemma", Journal of Economics Behavior and Organization, Vol.29, pp.87-112, 1996.

[12] M. Mitchell, "An Introduction to Genetic Algorithms", MIT Press, 1998.

[13] D.E. Moriarty and R. Miikkulainen, "Discovering Complex Othello Strategies Through Evolutionary Neural Networks", Connection Science, Vol.7 No.3, pp. 195-209, 1995.

[14] M.A. Potter and K.A. De Jong, "A Cooperative Coevolutionary Approach to Function Optimization", in the proceedings of 3rd Parallel Problem Solvign From Nature, pp. 249-257, 1994.

[15] M.A. Potter, K.A. De jong and J.J. Grefenstette, "A Coevolutionary Approach to Learning Sequential Decision Rules", in the proceedings of the 6th International Conference on Genetic Algorithms, pp.366-372, 1995.

[16] M. Riedmiller and B. Heinrich, "A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm", in the proceedings of IEEE International conference on Neural Networks, pp.586-591, 1993.

[17] D.E. Rumelhart, J.L. McMlelland and the PDP Research Group, "Parallel Distributed Processing", Exploration in the Microstructure of Cognition, Vol. 1, MIT Press, 1986.

[18] T.P. Runarsson and S. Lucas, "Co-evolution versus Self-play Temporal Difference Learning for Acquiring Position Evaluation in Small-Board Go", IEEE Transaction on Evolutionary Computation: Special Issue on Evolutionary Computation and Games, Vol.9, pp. 628-640, 2005.

[19] R.E. Smith and B. Gray, "Co-Adaptive Genetic Algorithms: An Example in Othello Strategy", in the proceeding of The Florida Artificial Intelligence Research Symposium, 1994.

[20] A.L. Samuel, "Some Studies in Machine Learning using the Game of checkers", IBM Research and Development Journal, pp. 211-229, 1959.

[21] W. Schiffmann, M. Joost and R. Werner, "Optimization of the Backpropagation Algorithm for Training Multilayer Perceptrons", Technical Report, Second edition, University of Koblenz, Institute of Physics, 1993.

[22] G.J. Tesauro, "Temporal Difference Learning and TD-Gammon", Communications of the ACM, Vol. 38, No. 3, pp. 56-68, 1995.

[23] X. Yao and Y. Liu, "Fast Evolutionary Programming", in the proceedings of 5th annaual conference on Evolutionary programming, pp. 451-460, 1996.

[24] M. Buro, "The Othello Match of the Year: Takeshi Murakami vs. Logistello", ICCA Journal, Vol. 20, No.3, pp.189-193, 1997.

[25] X. Yao, "Evolving Artificial Neural Networks", in the proceedings of the IEEE, Vol. 87, No.9, pp.1423-1446, 1999.

[26] G.J. Tesauro and T.J. Sejnowski, "A Parallel Network that Learns to Play Backgammon", Artificial Intelligence, Vol. 39, pp.357-390, 1989.