In *Coevolution: Turning Adaptive Algorithms upon Themselves,*
Birds-of-a-Feather Workshop, Genetic and Evolutionary Computation Conference
(Gecco-2001, San Francisco), 2001.

Co-Evolving a Go-Playing Neural Network

Alex Lubberts
Department of Computer Science
Twente University
Enschede, The Netherlands
d.a.lubberts@student.utwente.nl

Risto Miikkulainen
Department of Computer Science
The University of Texas at Austin
Austin, TX 78712
risto@cs.utexas.edu

Abstract

When evolving a game-playing neural network, fitness is usually measured by playing against existing opponents. In this article that need is overcome by applying the competitive co-evolutionary techniques of *competitive fitness sharing*, *shared sampling* and *hall of fame* to the SANE neuro-evolution method. This approach is tested by evolving networks to play go on small boards. Networks evolved through co-evolution were compared against those evolved against the gnugo program. The co-evolutionary techniques were found to speed up the evolution and to result in better networks not limited by the quality of the opponent.

1 INTRODUCTION

When evolving a network to perform a complicated task such as game playing, one needs to have an opponent to measure the fitness of a network. For simple tasks this does not pose a problem, but when the task is more difficult, like playing the game of go, it does, because there are no go programs that play at a significant level.

In this article we will discuss the possibility of removing the need of having a well performing opponent by using competitive co-evolution. One way to define competitive co-evolution is by evolving two populations: one is a population of *hosts* that try to find an optimal solution, the other is a population of *parasites* that, instead of trying to find an optimal solution, try to defeat the hosts by making use of their weaknesses (cf. Rosin, 1997). The fitness of the hosts is calculated using *competitive fitness sharing*. To reduce the number of games that have to be played, *shared sampling* is used. To make sure that the absolute quality of the hosts is increasing, they will not only be tested against a subset of parasites, but also against the best hosts of previous generations (i.e. a *hall of fame*, Rosin, 1997). As the evolutionary technique we will use the SANE neuro-evolution method (Moriarty and Miikkulainen, 1997), because it has been shown effective in the go domain (Richards et al., 1998). The results show that the learning speed is increased by using the co-evolutionary techniques and the level of play is not limited by existing opponents.

2 THE GAME OF GO

The game of go is an ancient board game which is believed to have originated in China. The rules of the game are relatively simple, yet there are no go-playing computer programs that play at a significant level.

Go is played on a board with a grid consisting of 19 horizontal and 19 vertical lines. Two players, black and white, take turns in placing stones on empty intersections of the grid, starting with the black player. Instead of placing a stone, a player is also allowed to pass. When three passes are executed in a row, the game ends and the score is calculated.

Stones of the same colour that are adjacent to each other are called a group. If a group is horizontally or vertically adjacent to an empty intersection, that group is said to have a 'liberty'. If a group has no liberties, i.e. it is surrounded on all sides by enemy stones, the opponent has captured the group and all stones in the group are removed from the board. If it is impossible to keep a group of stones from being captured, it is not necessary for the opponent to capture it. The group stays on the board until the game is over and is then removed from the board as if it were captured. Such a group is said to be 'dead', as opposed to a group whose capture is not unavoidable which is said to be 'alive'. At the end of the game, the captured stones will be added to the opponent's score.

One is not allowed to reduce the number of liberties of one's own group to zero (which is called 'committing suicide'). Finally, the 'ko-rule' states that no board configuration may occur twice in a game.

There are two ways of calculating the score of a player:

Chinese scoring: The score is determined by counting the captured enemy stones and all empty intersections that are completely surrounded by only stones of that player.

Japanese scoring: The number of the player's own stones is added to the number of captured enemy stones and the surrounded empty intersections.

Chinese scoring is the most common; however, we will use Japanese scoring, because gnugo(the go program we use for evaluating the performance of the evolved networks) uses it as well. In order to give the players an equal chance of winning the game, the weaker player may be given some points in advance, called 'komi'. We will utilise komi in our experiments to make competition more even.

3 APPROACH

Richards et al. (1998) evolved a go playing network for a small board with good results. However, the level of play was limited by the level of the existing program used as a 'sparring partner'. What is the use of evolving further when you are already able to beat your opponent? This problem can be addressed by evolving two populations trying to beat each other. This way, theoretically, the network should always try to improve. We adapted the SANE neuro-evolution method to simultaneously evolve two populations that challenge each other.

3.1 CHANGES TO THE RULES OF GO

To make automated play easier, we made changes to two of the rules of the game of go:

- *The ko-rule.*

  Implementing this rule would require saving every previous board configuration and comparing those to the current configuration. Instead, we chose just to compare the current configuration to the configuration two moves ago. Such a check is easy to compute and still catches most of the ko situations.

- *It is unnecessary to capture stones that cannot be saved.*

  It is a difficult problem to determine which stones are dead and which stones are alive. Therefore, we assume that all stones that are left on the board are alive. So, a player has to explicitly capture the stones that she or he thinks are dead.

In addition to the above changes, we placed an upper bound on the number of moves. Once the number of moves in the game reaches that upper bound, a tie is declared. This is done in case the simple check for ko does not work and the game gets stuck in an infinite loop.

Evolving networks that play on a 19×19 board is not practical at the moment. First, many generations would be needed to reach a given level. Second, the networks would have to be quite big (there have to be a lot of neurons in the input and output layer and the hidden layer) and therefore the populations would have to be large (otherwise, one would end up with too many similar networks), which also increases the time needed. For this reason we chose to do most of the experiments on a 5×5 board. However, such a simplification changes the game drastically. When playing on a 19×19 board, the consequences of a move may be noticeable only after several other moves. With a 5×5 board all moves made are 'local', i.e. the consequences of a move are immediately noticeable. Still, even with the 5×5 board the basic go principles about life and death must be learned; also, during the end game on a 19×19 board all moves are local as well.

There is one complication that comes with the assumption that all stones on the board are alive in combination with the Japanese scoring we use: it is possible to reduce the score of the opponent by letting him fill the empty intersections by just 'throwing in' stones in his territory. However, on such small boards this will not be a problem.

3.2 SANE

SANE (Symbiotic Adaptive Neuro-Evolution; Moriarty and Miikkulainen, 1997; Richards et al. 1998) differs from standard evolutionary algorithms in that instead of evolving complete neural networks, a population of *neurons* and a population of *blueprints* (that specify which neurons to combine into a neural network) are evolved. By evolving neurons instead of complete networks, the search space is decomposed and groups of neurons are able to specialise on different parts of the task. This way, diversity is maintained and the algorithm does not get stuck on a suboptimal solution as easily as other neuro-evolution methods (Moriarty and Miikkulainen, 1997). The blueprint population then searches for effective combinations of neurons.

Each individual in the neuron population specifies a neuron in the hidden layer of a two-layer neural network as a set of weighted connections made from the input layer or to the output layer as shown in figure 1. The first number in the genome indicates which neuron (in the input or output layer) to connect to and the second number specifies the weight of the connection. The third number again indicates a neuron, and so on. The number of connections

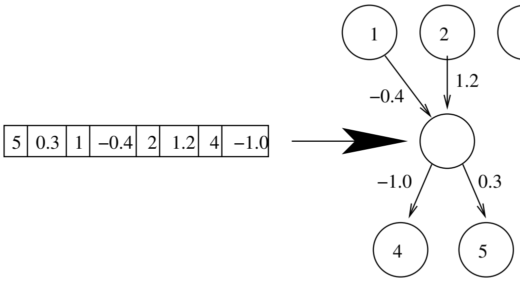

Figure 1: An individual from the neuron population. The genome (on the left) specifies which connections are to be made and what weight they have.

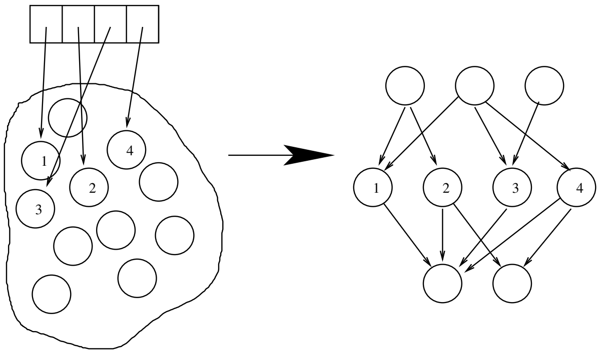

Figure 2: An individual from the blueprint population. The genome (the bar) points to neurons in the neuron population of which the hidden layer is made up.

each neuron has is fixed, but they are arbitrarily distributed among the input and output layer.

A *blueprint* genome consists of pointers to neurons in the neuron population to be used in a network (figure 2), so that efficient combinations of neurons are maintained. Encoding in this way encourages neurons in the neuron population to specialise and keeps well-performing neurons from being discarded just because they were evaluated in an ineffective combination with other neurons.

One iteration in the evolution process is divided into two steps: an evaluation phase and a reproduction phase. During the evaluation phase, each blueprint is used to build a network whose performance is evaluated and is assigned a fitness value. Each neuron receives a fitness value that is equal to the summed fitness values of the best five networks in which it participated. In the reproduction phase, both populations are ranked based on fitness. For each elite individual, a mate — which is also an elite individual — is chosen and a one-point crossover is used to produce two offspring. The offspring replaces the worst performing individuals in the population. Finally, mutation, with a chance of 0.5%, is performed on both the population of neurons and the population of blueprints. This way, SANE performs two evolutionary searches simultaneously: a search for neurons that implement useful sub-tasks, and a search for effective combinations of these sub-tasks.

SANE has been shown effective in several sequential decision tasks such as playing go (Richards et al., 1998) and Othello (Moriarty and Miikkulainen, 1995), and controlling a robot arm and a mobile robot (Moriarty and Miikkulainen, 1996). It was therefore selected as the starting point for the co-evolution experiments reported in this paper.

3.3 COMPETITIVE CO-EVOLUTION

Co-evolution is the simultaneous evolution of two or more populations with a common fitness landscape. Competitive co-evolution can be defined as the evolution of a *host* population from which individuals compete directly against individuals from a *parasite* population, which also evolves (Rosin, 1997; Rosin and Belew, 1995). Rosin (1997) defined the host population as the population that is currently being evaluated and the parasite population as the population the test cases come from. So, each population takes turn in being either the host or parasite population in this definition.

Since both populations are evolving simultaneously and a success in a competition for an individual from one population means a failure for an individual from the other population, both populations challenge each other, which can lead to an 'arms race'. Hopefully, this results in an ever-increasing performance, approaching the optimal solution. However, the cases on which the hosts are tested must neither be too easy, nor too difficult, otherwise the evolution will stagnate: which individuals should get the opportunity to reproduce in the case that all hosts win or all hosts lose to all parasites?

Rosin and Belew (1995) describe *competitive fitness sharing*, *shared sampling* and *hall of fame* as techniques for competitive co-evolution. These techniques are used to implement competitive co-evolution in our experiments as well.

3.3.1 Competitive Fitness Sharing

Usually, the fitness value of an individual is based on the score it acquired during its evaluation (e.g. a binary value indicating the success/failure of the individual). In *fitness sharing* (Holland, 1975; Goldberg and Richardson, 1987), similarity is taken into account in such a way that unusual individuals get rewarded. *Competitive fitness sharing* (Rosin and Belew, 1995) takes similarity into account by rewarding individuals that beat opponents few others can, even though the individual might not beat as many opponents as others can.

The fitness f_i assigned to an individual i is:

$$f_i = \sum_{j \in O} \frac{1}{N_j}$$

where O is the set of opponents defeated by i and N_j the number of times an opponent j lost.

The consequence of using competitive fitness sharing is that an individual that is the only one that can defeat a certain opponent is saved from extinction since it receives a large reward for doing so. This way, diversity in the population is encouraged. The information this individual contains will be spread throughout the population so that individuals from later generations all are able to defeat that opponent.

3.3.2 Shared Sampling

Shared sampling (Rosin and Belew, 1995) provides a mechanism to reduce the total number of competitions that have to be played by selecting a sample set from the population. Of course, this could be done by selecting a random subset from the population, but when there is more information about the population available, it might as well be used to select a more diverse and representative sample. This way, all, or at least more, types of opponents will be challenged. The resulting sample set makes a good set of opponents to the other population.

The way to choose such a teaching set resembles competitive fitness sharing. First, the individual that defeated most opponents during the previous generation is selected. Then, the individual that competed well against opponents against which the first one did not compete well is selected, and so on, until the set is deemed sufficiently large.

3.3.3 Hall of Fame

To ensure that individuals do not lose the ability to defeat opponents from previous generations, *hall of fame* (Rosin, 1997) is introduced. Hall of fame preserves the best opponent from each previous generation, that is, the individual that had the most number of successes (instead of the individual with the highest fitness). In addition to the opponents from the current generation, each individual is tested against a sample from the hall of fame. Rosin (1995) applied shared sampling to the hall of fame, but found that selecting a random subset from the hall of fame works just as well. Therefore, in our experiments a random subset from the hall of fame is used.

3.4 APPLYING COMPETITIVE CO-EVOLUTION TO SANE

We use different definitions for hosts and parasites than Rosin (Rosin, 1997), perhaps closer to the biological terms. We define a *host* as an individual that tries to find an optimal solution to the problem, whereas a *parasite*, rather than finding a general solution, tries to defeat the hosts by making use of their weaknesses. The parasites can be seen as a set of test cases for the hosts.

From a biological point of view, a host has to try to offer resistance to as many parasites as it can to increase its chance of survival. For a parasite to survive, it is sufficient to just be able to make use of the weakness of one host only. We have tried to mimic this behaviour by applying the techniques described above differently to the host and parasite population. This way, a diverse parasite population should result, challenging the hosts to develop a general playing style, defeating as many parasites as they can. In order to encourage the hosts to develop generality, the parasites have to be diverse. To make sure the difference in playing level is not too large, komi, with one point at a time, is granted to the weaker player.

4 EXPERIMENTAL SETUP

4.1 THE NETWORK

In the experiments we used the population and networks sizes found to be effective by Richards et al. (1998), namely a neuron population of 2000 neurons, a population of 200 blueprints, and a network size of 100 neurons in the hidden layer. The input layer consists of two input units for each intersection on the board. The first input unit is activated if a black stone occupies the intersection. The second input unit is activated if a white stone is present on the intersection. The output layer consists of one output unit per intersection. The output is a value between 0 and 1 and indicates how good it is to play on that intersection: the higher the value, the better.

First, the board position is fed into the neural network. Then a move is made on that intersection that received the highest value among all legal moves. If all values are below 0.5, the network passes.

4.2 THE RUNS

For the experiments, a run consists of 250 generations. For each generation, the number of games won by the hosts and by the parasites during the evolution will be administered to see if komi should be granted. The best host and the best parasite from each generation are saved. When evaluation is over, a tournament will be held; each host (from the set of saved networks) will play against each parasite. The number of wins for each host will be counted and should increase over the generations.

To measure the absolute quality of the evolved networks is difficult. One possible reference point is gnugo, a publicly-available rule-based go program. However, all networks defeated gnugo after very few generations. Gnugo is apparently not designed to play on such small boards. However, it seems that the networks have learned something about life and death: because Japanese scoring is used, it does not hurt to place stones in ones own territory, as long as the group of stones stay alive; most networks place stones in their own territory, but most stop just in time to stay alive. Although it is still relatively easy for a human player to beat the networks, they generally make reasonable go moves.

This observation was quantified experimentally in three different ways:

1. The performance of the co-evolved networks were compared to those evolved against gnugo.

2. The performance of hosts and parasites was analysed over time.

3. The evolution speed of the system was compared to ablated versions where fitness sharing, shared sampling, and hall of fame were each turned of at a time.

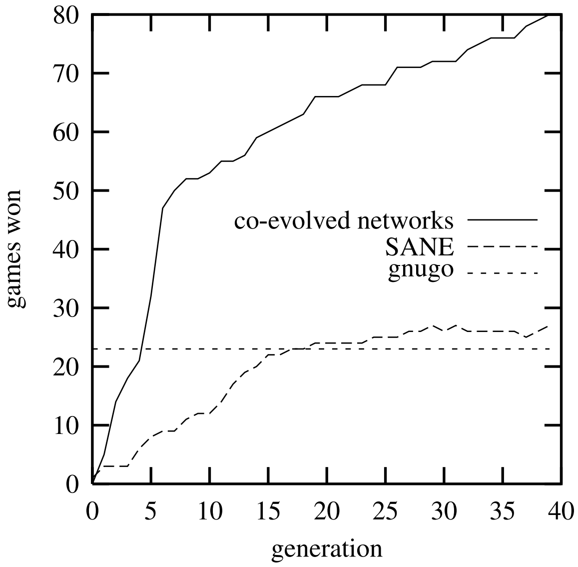

Figure 3: The outcome of the tournament between gnugo, the co-evolved networks and the networks evolved against gnugo. Plotted is the number of games each network won. Co-evolution achieves higher levels of play than standard evolution, which stagnates just slightly above the level of the opponent.

5 RESULTS

5.1 COMPARISON

The most important result is that co-evolution indeed allows better game-playing behaviour than standard evolution against a fixed opponent. Using SANE, 40 generations of networks were evolved against gnugo. A tournament was then run between these networks, gnugo, and the first 40 generations of the co-evolved networks from a typical run. The number of games each network and gnugo won is plotted in figure 3.

Gnugo won 23 games in this tournament: it took the co-evolved networks 5 generations to beat gnugo while the networks evolved against gnugo took 18 generations. The big leap in number of games won by the co-evolved networks between the 5th and 7th generation indicate the point from where the co-evolved networks were able to beat all the networks evolved against gnugo. The almost flat line after generation 18 of the networks evolved against gnugo shows that evolution stops after reaching the goal of beating gnugo as opposed to the increasing number of games won by the co-evolved networks. This indicates that the networks evolved against gnugo are limited by the level of play of existing opponents, whereas the co-evolved networks keep improving.

5.2 A TYPICAL RUN

Figure 4 illustrates the progress of a typical co-evolution run. A dot indicates that the best host network of the generation corresponding to that horizontal coordinate beat the best parasite network corresponding to that vertical coordinate (no dot indicates a loss or a tie). Ideally, the lower right half of the plot would be filled with dots, which would indicate that the hosts (only) win from parasites of previous generations. As can be seen, there are more dots in the lower right part. The number of dots in each column increases over time, which indicates that something is being learned. This particular run was able to defeat gnugo after just 5 generations, so there appears to be considerable progress even after that reference point.

5.3 ABLATION STUDIES

To verify that each co-evolutionary technique does indeed contribute to the performance, each of them were removed from the system in turn and a new set of simulations were run. The average number of generations it took each test to defeat gnugo was then measured (table 1). All ablations were able to defeat gnugo. However, the full system reached this level several times faster than the ablated versions. All techniques therefore contribute to the performance of co-evolution.

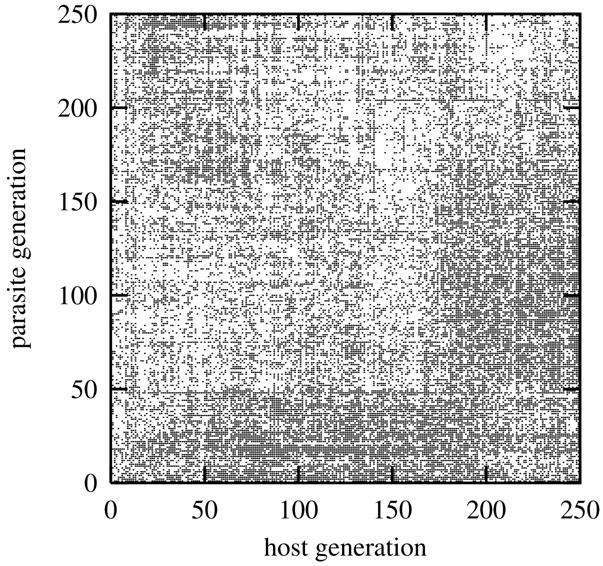

Figure 4: A typical co-evolution run. A dot indicates that the best host network of the generation corresponding to that horizontal coordinate beat the best parasite network corresponding to that vertical coordinate. There are more dots in the lower right, indicating that the level of play is improving.

technique switched off	average number of generations
fitness sharing	29
shared sampling	20
hall of fame	12
none	7

Table 1: The average number of generations needed to defeat gnugo.

6 CONCLUSION

The results show that co-evolutionary techniques speed up SANE and develop a level of play not limited by available opponents. This result is a good start; however, whether such techniques will eventually allow playing on full size boards is an open question at this point. We believe that other advances will be necessary, such as more structured representations of the game states, but co-evolutionary techniques are likely to be part of the solution.

References

Goldberg, D. and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 148–154, San Francisco, CA. Kaufmann.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. University of Michigan Press, Ann Arbor, MI.

Iwamoto, K. (1977). *Go for Beginners*. Random House.

Moriarty, D. and Miikkulainen, R. (1995). Discovering complex othello strategies through evolutionary neural networks. *Connection Science*, 7:195–209.

Moriarty, D. and Miikkulainen, R. (1996). Evolving obstacle avoidance behavior in a robot arm. In Maes, P., Mataric, M., Meyer, J.-A., and Pollack, J., editors, *From Animals to Animats: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, pages 468–475, Cambridge, MA. MIT Press.

Moriarty, D. and Miikkulainen, R. (1997). Forming neural networks through efficient and adaptive coevolution. *Evolutionary Computation*, 5:373–399.

Richards, N., Moriarty, D., and Miikkulainen, R. (1998). Evolving neural networks to play go. *Applied Intelligence*, 8:85–96.

Rosin, C. (1997). Coevolutionary search among adversaries. Master's thesis, University of California, San Diego.

Rosin, C. and Belew, R. (1995). Methods for competitive co-evolution: Finding opponents worth beating. In *Proceedings of the Sixth International Conference on Genetic Algorithms*.