Coevolution of Neural Go Players in a Cultural Environment

Helmut A. Mayer
Department of Scientific Computing
University of Salzburg
A-5020 Salzburg, AUSTRIA
helmut@cosy.sbg.ac.at

Peter Maier
Department of Scientific Computing
University of Salzburg
A-5020 Salzburg, AUSTRIA
pmaier@cosy.sbg.ac.at

Abstract- We present experiments (co)evolving Go players based on artificial neural networks (ANNs) for a 5x5 board. ANN structure and weights are encoded in multi–chromosomal genotypes. In evolutionary scenarios a population of generalized multi–layer perceptrons (GMLPs) has to compete with a single Go program from a set of three players of different quality. Two coevolutionary approaches, namely, a dynamically growing culture, and a fixed–size elite represent the changing environment of the coevolving population. The playing quality of the (co)evolved players is measured by a strength value derived from games against the set of three programs. We also report on first experiments employing recurrent networks, which allow a direct structural representation of the Go board. Finally, the quality of all the best (co)evolved players is evaluated in a round robin tournament.

1 Introduction

With the advent of computers, board games have attracted many researchers, e.g., [1], as the computational intelligence of game playing programs can be directly related to the intelligence of its human opponent. Out of all board games, chess has received the most attention with efforts beating the human world champion finally being successful in 1997. (*Deep Blue*, a chess–playing IBM supercomputer, defeated Garry Kasparov, the reigning world champion in chess [1]).

The board game Go has received increasing attention in recent years, as unlike chess programs the best Go programs are still at a mediocre amateur level, i.e., a good amateur Go player easily beats the machine. The rule set of Go is very small, but the seemingly simple concepts build into deep and complex structures on the board. For an excellent and compact introduction we refer to [2]. Despite the simplicity of Go's rules, the game's strategies and tactics are difficult to put into analytical or algorithmical form. There are mainly three reasons why Go is hard for traditional computer game–playing techniques.

First, the number of possible moves (the branching factor) in the majority of game situations is much larger than in games such as chess or backgammon with about 20 legal moves for each board position. On a standard 19×19 Go board a player has the choice among 200–300 potential moves. Hence, in a common game tree representation, where each node is associated with a board situation and each branch with a move, the number of nodes grow exponentially with a base of 200. A Go computer program playing with a very moderate tree depth of four had to evaluate 10,000 times the number of moves a chess program has to ponder.

Second, Go is a game of mutual dependent objectives. While in chess the goal is very explicit (capture of the opponent's king), in Go the aim of securing territory (where each board intersection counts as a point) can be achieved by capturing opponent's stones (death) as well as by securing own stones (life). As a consequence, evaluation functions precisely assessing a board situation can hardly be defined, as human expert players often rely on rather intuitive concepts, e.g., *good* and *bad shape* (of stones). Hence, ANNs having been successfully applied in the field of pattern recognition are promising candidates to improve the quality of Go programs.

Third, though Go has been played for thousands of years in China and Japan, the first professional Go players started to earn prize money 45 years ago. Professional chess has a tradition of 130 years resulting in much more literature on opening, mid–, and end game theory based on millions of recorded games played by expert players. As a matter of fact, today's extremely strong chess programs rely on human expertise to defeat human expertise.

A radically different approach is the construction of computer players by means of *Evolutionary Computation* (EC). Here, an initial number of (often random) players (programs) play against each other, the winners survive, and exchange and randomly alter (mutate) parts of their genetic material (the program code) so as to produce new programs undergoing the same evolutionary procedures. Eventually, the programs improve their playing strength without any explicit incorporation of a priori knowledge, which gives these systems the potential to "invent" game strategies no human player has ever discovered.

Moriarty and Miikulainen (1995) presented the evolution of neural networks playing the game of Othello. The fitness of the ANN players has been evaluated by a random player and a program employing α-β search. Evolved players could easily beat the random player (after 100 generations), and could also win against the program (after 2000 generations), which adhered to a popular Othello strategy. A more complex strategy used by human expert players has intentionally not been integrated into the programmed player. It could be shown that evolution discovered the novel (counter–intuitive) strategy so as to beat the α-β program [3].

[1] http://www.research.ibm.com/deepblue/

Chellapilla and D. B. Fogel (2001) presented an evolved ANN playing the game of checkers. The value of the single output neuron was used as an evaluation of the current board situation presented to the input layer. The board evaluation has been utilized to perform α–β search with a (standard) search depth of four. After 840 generations (six months) the best network has been evaluated by games against human players. A checkers rating system allowed to categorize the performance of the network. The neural player achieved *Expert* level (third best category) and could even achieve a win against a higher rated human expert player [4].

The "star" among artificial board game players is Tesauro's (1995) neural backgammon player *TD–Gammon*. Based on *Temporal Difference* (TD) learning, a reinforcement learning technique, a network has been trained in self–play by only receiving feedback on the outcome of games. After millions of training games (in its latest version) TD–Gammon is estimated to play at a level extremely close to the world's best human players [5].

2 (Co)Evolution of Neural Go Players

The automatic generation of game–playing ANNs by artificial evolution offers some appealing advantages to conventional ANN training. Even, if training yields an ANN player having extracted all the concepts hidden in the training data, it is very likely that it will never surpass the strength of the players, whose games constituted the training data. E.g., in [6] ANNs having been trained with chess games by master players, played reasonably against strong players, but failed to beat weak players.

Evolution of game playing ANNs does not require any knowledge of the game, but only the games' rules and the feedback about the outcome of the game. Hence, in theory the evolved neural player could have playing abilities beyond any human player, as it does not rely on human expertise at all. Nice as this may sound, there are practical limitations to ANN evolution, most prominently, the computational cost associated with the evolutionary process, where thousands and millions of individuals (neural players) have to be evaluated. Hence, we restricted evolution of Go players to the simple 5×5 board, which is mostly used for educational purposes and demonstration of basic concepts of the game. Though, we carried out the experiments with the *netJEN* system (a pure Java application for ANN evolution) designed and implemented by the authors, which supports distributed computation, from our point of view evolution of Go players for a 9×9 board is the current limit (unless one spends months and years of CPU time).

2.1 ANN Board Representation

We have extensively experimented with a variety of different board representations, but in the end a simple representation also suggested in related work [7] turned out to be the best.

Each intersection on the Go board is represented by two input neurons, one for each player. A 1 indicates that the intersection is occupied by the corresponding player, a 0 that it is not, i.e., two zeros represent an empty intersection, and two ones are illegal. We rather speak of two players instead of black and white, as the same network may play both colors (even against itself) by simply discerning between own stones and opponent stones. At the output layer each neuron is simply assigned to an intersection. The move corresponding to the highest activation is selected. If this move is illegal, e.g., the intersection is occupied, the move with the next highest activation is chosen. These representations result in 50 input and 26 output neurons (including the pass move) for the 5×5 board .

2.2 ANN Encoding and Genetic Operators

ANN evolution is based on a direct encoding scheme generating *Generalized Multi–Layer Perceptrons* (GMLPs), which have no defined layered structure between input and output layer, and may contain any forward connections between neurons (including direct connections from input to output neurons). The number of hidden neurons, the connections, and the connection weights are evolved on separate chromosomes, hence, the complete ANN genotype consists of three chromosomes. During recombination the chromosomes of two parents are shuffled (exchanged) with a shuffle rate $p_s = 0.5$ [8]. The multi–chromosomal encoding enables the use of different encodings (and corresponding operators) on different chromosomes: the hidden neurons, and the connections are encoded by bitstrings (*Genetic Algorithm* style), while the weights are encoded by real numbers (*Evolution Strategies* style).

Each hidden neuron and each connection is represented by a single bit (*Marker*) in the corresponding chromosomes. The markers are a simple analogue to activators/repressors regulating the expression of wild–type genes. A hidden neuron/connection marker determines, if the specific neuron/connection associated with it is present in the decoded network. The maximum number of hidden neurons (neuron markers) has to be set in advance, hence, this evolution technique could be labeled as *Evolutionary Pruning*, since the system imposes an upper bound on the complexity of the network.

The mutation operator for the binary chromosomes and the real number chromsome is the standard bit flip mutation, and σ–self–adaption (σ–mutation) [9], respectively. With σ–mutation each object parameter x_i (here a connection weight) has an associated strategy parameter σ_i controlling mutation of the object parameter as given by

$$x_i' = x_i + \sigma_i' \cdot N(0,1), \tag{1}$$

where x_i' is the mutated object parameter, and $N(0,1)$ the normal distribution. The strategy parameters σ_i are mutated according to

$$\sigma_i' = \sigma_i \cdot e^{(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1))}, \tag{2}$$

with $\tau' = \left(\sqrt{2n}\right)^{-1}$, $\tau = \left(\sqrt{2\sqrt{n}}\right)^{-1}$, n being the number of object parameters, and $N_i(0,1)$ indicating that a new random number is drawn from the distribution for each

strategy parameter. A simplified form of σ–mutation only uses a single strategy parameter for mutations of the object parameters (termed *single σ–mutation* in the following).

The recombination operator for all chromosomes is 2–point crossover (occuring separately on each chromosome), and the selection method of choice is *Binary Tournament* selection with replacement.

2.3 Coevolution

Coevolution can be *Competitive (Parasitic)* or *Cooperative (Symbiotic)*. In parasitic coevolution the host population(s) receive a fitness directly proportional to the fitness of the solution, while the parasite population(s) receive an inversely proportional fitness, i.e., the more the parasites harm the host(s), the more fitness they receive. With symbiotic coevolution all populations receive a fitness proportional to their collaborative success in solving a problem.

Theoretically, coevolution allows "open–ended" evolution, i.e., the only limit for the quality of a solution is the evolutionary time (number of generations). However, a few problems have been identified with coevolution [10] leading to stagnation of the coevolutionary progress. Amongst them are *Super Populations* dominating other populations, the *Moving Target* problem introducing (too much) noise in fitness evaluation, and the occurrence of cycles. Another potential problem of coevolution, specifically, in the context of game play, is that good players often lose the ability to defeat weak players. When all or most weak individuals have been sorted out, the remaining good players are no longer forced to beat weak players.

As a consequence to these known problems and inspired by the concept of *Cultural Algorithms* [11] we devised the simple, but intuitively appealing strategy of *Cultural Coevolution* presented in the next section.

2.3.1 Cultural Coevolution

In cultural coevolution a population evolves based on its own culture, i.e., individuals of former generations preserve and accumulate knowledge being available for the current generation. However, in our simple model of evolution knowledge cannot be transferred explicitly, but individuals that brought new knowledge to the culture become part of the culture. In the context of neural Go players, the culture is a collection of master players being the dynamically growing number of opponents (but also teachers) of the evolving population.

In more technical terms, coevolution starts with a random start population and an empty culture (no Go knowledge). To get things going a random member of the start population is added to the culture. Though, this certainly is no master player it resembles the current cultural knowledge. Each individual plays two games (black and white) against each player in the culture. The individual's fitness is determined by the overall win rate. If an individual wins all the games against the master players, it becomes a master itself and is added to the culture.

In order to analyze, if cultural coevolution has a measurable effect, when compared to more conventional coevolution techniques, we devised a technique presumably introducing some of the coevolutionary pitfalls discussed above. The *Elite* is a fixed number of players resembling only the best players having emerged during evolution. The fitness of an individual in the evolving population is determined in two games against each elite player. If an individual wins all games against a specific elite player, the individual replaces the elite player. When starting an elite coevolution, the elite is filled randomly with players from the start population.

An approach similar to cultural coevolution is the *Hall of Fame* concept [10] utilized in [12]. In the original procedure the best individual of each generation is added to the hall of fame. As this technique may lead to a large number of individuals in the hall of fame (depending on the number of generations), individuals of the current population competed against a random hall of fame subset in [12]. The main difference to cultural coevolution is the possibly large number of neural players with similar capabilities in the hall of fame, which, again, may lead to focussing on specific playing strategies in the course of coevolution.

2.4 Performance Measures

In order to monitor the development of the (co)evolving go players we devised the following performance measures.

We define the strength $s = \frac{w}{g}$, as the win rate (w is the number of wins) of a player challenging one or more Go players in a number of games g. In the following experiments (Section 4) the strength has been measured in games against three computer players (Section 3) of different quality ranging from a pure random player to a heuristic player including search for common Go patterns on the board.

The strength value of an ANN player does not indicate to which degree the network "understands" the game. A basic indicator of game comprehension is the number of illegal moves a network tries to play. Consequently, we defined the competence C measuring the ability of a neural player to distinguish between legal and illegal moves as

$$C = \frac{1}{n} \sum_{i=0}^{n} 1 - \frac{t_i}{p_i} \tag{3}$$

For each of n games the ratio of all illegal moves t_i tried in a game to the number of all illegal moves p_i possible generates the competence's raw value. An ANN player with $C = 1.0$ did not select a single illegal move in all n games, whereas a player with $C = 0.0$ always tried all illegal moves before it placed its stone correctly. A competence of 0.85 indicates that on average the neural player intended to play 15% of all possible illegal moves but avoided all others.

3 Computer Go Players

For the (co)evolution of neural Go players and their evaluation we utilized three heuristic computer players of different playing abilities, which are briefly described in the following.

The *Random* player's only "knowledge" of the game is the ability to discern between legal and illegal moves, i.e.,

out of all legal moves (including the pass move) one is chosen randomly with uniform probability distribution. This player's main purpose is to detect very basic Go skills in a computer player, as a human novice with some hours of Go practice should easily beat the Random player. Also, it serves as a test for a neural player that possibly is able to win against a modest computer player, but does not have a general concept of Go, i.e., it may lose against Random.

The *Naive* player may be compared to a human knowing the rules of Go, and having played some games is familiar with basic concepts. It is able to save and capture stones, and knows when stones are definitely lost. Weak stones, i.e., stones in danger of being captured, are saved by connecting them to a larger group, so that a weak stone becomes a member of a living group (or at least of one with more liberties).

JaGo is a Go program written in Java [2] by Fuming Wang. JaGo is the best computer player we have used. It knows standard Go playing techniques (saving and capturing stones), and searches the board for 32 well–known Go patterns and its symmetrical transformations. A few minor program errors have been fixed, and time performance has been increased in some parts by the authors.

In order to rate a Go player's strength there are ranking systems for amateur and professional players. The amateur ranking system starts with the student (*kyu*) ranks from 35 kyu up to 1 kyu (best). When an amateur becomes a master (*dan*) player, she gets the rank of 1 dan (best is 7 dan). Professional ranks being above all amateur ranks are on a scale from 1 to 9 dan.

GNU Go [3] is a free Go program being able to play games on 5×5 to 19×19 boards. We used GNU Go 3.2 to determine the strength of JaGo, and arrived at a rank of about 16 kyu. GNU Go's rating is slightly better than 10 kyu on the *No Name Go Server* [4] (as of June 1, 2003), which corresponds to an advanced amateur player's capabilities on a 19×19 board.

Recently, Go on a 5×5 board has been solved [13]. Black wins with a score of 25 points (no komi), when playing the optimal opening move C3 (board center). Black also wins starting play with C2, C4, B3, and D3 (by a score of 3, no komi), however, with a komi of 5.5 these games are lost.

GNU Go optimally opens a game (C3) with the black stones on a 5×5 board, and passes correctly in reaction to black C3 playing the white stones. However, it also passes after black B3, C2, C4, or D3, but with optimal play could win with a score of 2.5. As an evolved ANN only would have to learn the correct opening move, GNU Go has not been utilized in evolution experiments, however, it definitely is an interesting evolution opponent on larger boards.

4 Experiments

This section presents experiments (co)evolving neural Go players employing feed–forward and recurrent ANNs.

[2] `http://www.cs.vu.nl/~jbmarkes/jago/`

[3] `http://www.gnu.org/software/gnugo/`

[4] `http://nngs.cosmic.org/`

4.1 Experimental Setup

In all experiments games are conducted on a 5x5 board with a komi of 5.5 for the white player. Evolution (Section 2) is taking place in a population of 50 individuals, which initially are created by random. The alleles of the two bitstring chromosomes, representing the hidden neurons and all connections, are set according to a probability randomly chosen for each individual (biased coin). The random values for the initial real number chromosome are drawn from the interval $[-0.1, 0.1]$.

The maximal GMLP network consists of 50 input, 20 hidden, and 26 output neurons corresponding to a maximum of $3,010$ connections. This transfers to a length of the real number chromosome of $3,056$ encoding the connection weights and 46 bias values for hidden and output neurons.

The structure of the recurrent ANN is composed of 25 input and 26 output neurons, which are fully connected (including self–connections) resulting in $2,601$ connections. The board situation is encoded by a value for each intersection (black = -1, empty = 0, white = 1), which is fed into the input layer via the neurons' bias. As the number of neurons is not evolved, the genotype consists of two chromosomes, a bitstring chromosomes with a length of $2,601$ encoding the connections, and a real number chromosome of length $2,652$ encoding the weights and biases.

For both structures the mutation rates of the binary chromosomes are set to $\frac{1}{l}$, where l is the chromosome length. σ–mutation with an initial $\sigma = 0.02$ is used for the weight chromosome. All neurons employ the sigmoidal activation function.

The playing quality of the (co)evolved ANNs is evaluated by their strength s, which is computed by playing $2,000$ games ($1,000$ with each color) against each of the three computer players Random, Naive, and JaGo (Section 3).

4.2 Evolution Experiments

In this section we describe experiments in which the ANNs have been evolved by playing against each of the dedicated computer players Random, Naive, and JaGo. Each experiment has been repeated 20 times. The fitness of an ANN is evaluated by the win percentage after playing a number of games (with both colors) against the fixed opponent. The maximal number of generations is $3,000$, but evolution is halted, when a neural player wins all games against its opponent, as this ANN is of maximal fitness.

4.2.1 Evolution versus Random

During evolution each ANN has to play 64 games against Random. Nearly all of them won more than 90% of games against the Random player. The strongest reached a win rate of 0.9545, the weakest a value of 0.8820. The strategies developed by the ANNs to defeat Random do not work well against the Naive player. The best ANN achieved a win rate of 0.2695, while the worst reached 0.0285. Not surprisingly, the evolved ANNs are not able to keep up with

JaGo. Except for two ANNs that reached a win rate around 0.08 all others played below 0.04. All of the ANNs reached a similar competence value in the range of 0.45 up to 0.50. The low competence is due to stubborn attempts to place stones in the board center even though the intersection may be occupied.

Only three evolved ANNs open a game with the optimal move C3 (Section 3). The ANNs rather place their first stones anywhere on the board, except the corners and the middle of the edges (A3, C1, C5, and E3). This reflects the obvious fact that Random is not able to capitalize on weak opening moves.

4.2.2 Evolution versus Naive

In the next experiments Random is replaced by the stronger Naive player. Again, the fitness of each neural player is assessed in 64 games. The best ANNs (single best of each run) evolved against Naive have a strength ranging from 0.48 to 0.69. The ANN with lowest strength (0.4815) achieved a win rate of 0.8205 against Naive being the fifth best win rate of the evolved ANNs. The moderate strength results from low win rates against Random (0.6030) and JaGo (0.0210) indicating the ANN's specialization in defeating Naive.

The evolved ANNs place their stones in the board center, and try to keep them connected, which is the same basic strategy the ANNs evolved against Random performed. However, the Naive nets are slightly more reactive to specific moves of its opponent. 25% of the best evolved ANNs played the optimal opening move C3. Ten ANNs play around C3, while the remaining five ANNs play the edge of the board, which normally is a bad choice, but exploits a weakness of Naive. It immediately tries to capture this stone, which gives the net enough time to establish a good position in the center.

4.2.3 Evolution versus JaGo

The next challenger for evolution is JaGo, a fairly sophisticated player (Section 3), on average winning 90% and 81% of the games against Naive, playing black and white, respectively. As JaGo needs much more time than the weaker players to consider its moves, but also exhibits less random behavior, we reduced the number of games against each network from 64 to 32 (in 19 runs, one run halted due to technical problems).

In Figure 1 the development of the mean and best fitness, and the mean competence of a population of an evolutionary run employing JaGo as opponent is shown.

While a Naive population acquires a mean fitness of 0.6 within about 200 generations, in a JaGo population it takes about 1,000 generations to reach 0.4 leading to a mean fitness of approx. 0.55 in the last generation 3,000. Four evolution runs proliferated a network winning all 32 games against Jago.

The JaGo ANNs connect their center stones quickly, as otherwise Jago would win easily. Additionally, they sometimes play elsewhere (*tenuki*) sacrificing single stones in order to distract Jago. Similar to evolution versus Random

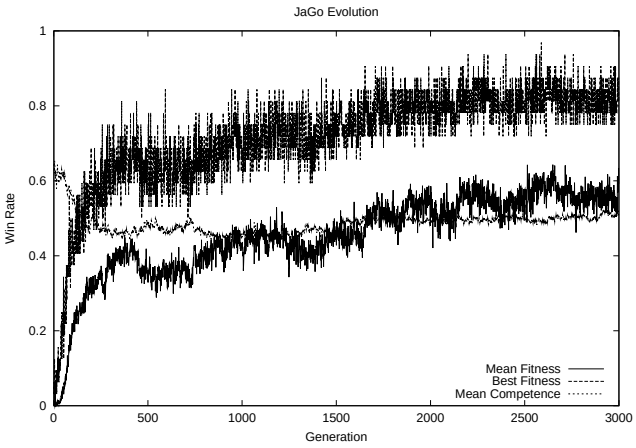

Figure 1: Fitness and competence statistics of evolving players against JaGo.

and Naive, the neural players often exhibit a preference to place their stones onto key intersections regardless of their state. The evolved ANNs have strength values ranging from 0.35 to 0.77. On average they defeat Random in 81%, Naive in 25%, and JaGo in 68% of games played. These win rates show that in this setting evolution generates specialists performing well against the single player they face during evolution, but fails to generalize. Specifically, one would expect that a net beating JaGo should easily beat the much weaker Naive.

Nine ANNs open the game optimally playing the first stone at the board center, and not a single neural player starts play at the edge. Remember that even though 25% of the Naive nets opened at an edge intersection they beat Naive in most cases. This indicates that evolution has adapted to the stronger play of JaGo. The strength of the evolved networks clearly corresponds to the opening move, as the nine nets playing C3 have an average strength of 0.6748, the seven nets playing B3 or D3 0.5276, and the the three remaining playing B2, C4, and D4 0.4248.

A main problem associated with the feed–forward structure and a simple board representation is that the information on neighborhood relations of intersections is not provided to the network. We could argue that most of evolution time is spent to acquire knowledge, which is initially available to a human seeing a Go board for the very first time. A fully connected input layer with recurrent connections (Section 4.1) gives evolution the possibility to represent board structure with network structure. Here, we performed only two runs, where each evolved recurrent net played 32 games against JaGo.

Compared to evolution of feed–forward ANNs (Figure 1) the population's fitness increases faster, and within 1,000 generations a recurrent network wins all 32 games against JaGo. The two star players play with a strength of 0.6927 and 0.6517. Though, these values are similar to the best evolved feed–forward ANNs, the recurrent players seem to have more general abilities, as the best recurrent net achieves higher win rates against Naive (0.4940) and Random (0.9465). Interestingly, the number of connections is

very similar in both star networks (1,296 and 1,294 out of maximally available 2,601).

Though, both evolved ANNs open the game at the optimal C3, they adhere to different strategies. One attacks enemy stones and defends its own stones, while the other tries to distract its opponent by playing the weak move A5 with the second stone.

4.3 Coevolution Experiments

Though, evolution generated neural players being able to defeat its single opponent faced during evolution, the above experiments also demonstrated the known problem of poor generalization of the evolved player, e.g., a network beating JaGo lost against the much weaker Naive. In order to improve the generalization capabilities we employed coevolutionary scenarios, where the networks never face a Go program representing human expertise, but only play against other coevolved networks. We compare the two coevolution approaches presented in Section 2.3, namely, cultural and elite coevolution.

4.3.1 Cultural Coevolution

In this experiment the fixed computer opponent is replaced by the dynamically growing culture. The fitness of each ANN in the population is the overall win rate of games against each culture net. Usually, two networks always play the same game against each other, hence, in most cases two games (with changing colors) are sufficient. The only exception are networks suggesting different moves for a board situation (identical maxima), which then are selected randomly resulting in different games. In this case a series of eight games is played. As no Go external knowledge is provided to the system, in a single run the number of generations had been increased to 55,000, which resulted in run times of up to twenty days. All other parameters are identical to the feed–forward setup described in Section 4.1.

In preliminary experiments we realized that evolution progress stagnated quickly due to saturated output neurons with the exact maximal value of 1.0. Increasing numbers of saturated output neurons turn the neural player into a random player, as moves are selected randomly out of those with a value of 1.0. In evolutionary settings these players are weeded out, as they do not succeed against a fair player. However, in the coevolutionary scenario the opponents are also likely to exhibit pseudo–randomness leading to stagnation. Thus, we switched to single σ–mutation, which seems to be less prone to above saturation effects.

Figure 2 shows statistical details of a cultural coevolution run.

Up to generation 10,000, 73 ANNs entered the culture, only five more nets were added, hereafter. The last culture net entered in generation 41,936 having a strength of 0.4513. Note that the mean fitness seems to drop slightly, however, with the culture growing dynamically, the evolved networks may face different cultures, hence, only phases of evolution without culture changes (addition of a neural player) can be compared exactly. The mean fitness of the population stays at a level of 0.85, although, the ANNs had to compete against the culture of growing quantity and quality. The competence of ~ 0.4 is considerably lower than in evolution against a computer player (~ 0.5). This can be attributed to the networks' missing ability to pass early in the end game, when few legal moves are left. As now two such players meet each other, the competence even drops.

Most culture nets consist of nine to 15 hidden neurons. Rarely, two successive masters have the same number of hidden neurons pointing out that cultural coevolution enables diversity. The number of connections increases slightly with improving culture, i.e., the ANNs become more complex.

In Figure 3 the graphical representation of the strength of each player in the culture is shown.

The strength increases steadily indicating that the culture ANNs' Go playing abilities become more sophisticated. The oldest net in the culture is able to win games against Random only, and expectedly, loses all games to Naive and JaGo. Up to master 18 the strength rises above 0.35, mainly because of improvements against Random. Subsequently, the strength increases due to wins against Naive and JaGo, and continuing success against Random, which nicely demonstrates that the culture does not lose the ability to beat the weakest player. Beginning with master 58 all players exhibit a strength above 0.4. From masters 36 to 48 a disproportional rise in win rates against JaGo can be observed. Analysis of these culture nets showed that, when playing the black stones, the nets often force JaGo into a trap, where it makes a bad mistake.

The opening moves of the 78 master nets are another indicator for the potential of the culture. The oldest 29 masters play various openings, but all others play the optimal move C3. Very weak openings (e.g., E4 by master 2, E2 by 3, pass by 4) have been discarded quickly beginning with master 5.

An example game of the youngest master 77 (beating all others) playing the black stones against JaGo is presented in Figure 4.

The first moves until 10 is a standard and correct opening of both sides. After 10 the move both players should

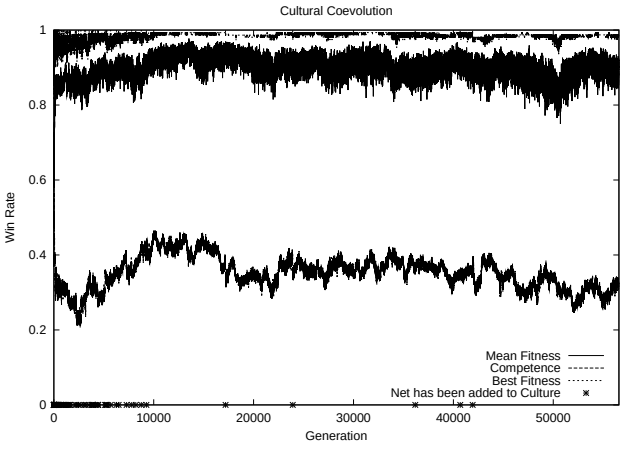

Figure 2: Mean and best fitness, competence, and culture additions in a cultural coevolution run.

prefer is 15, but Net plays 11 to create eye space. JaGo prepares an attack at 7 with 12, which results in death of white's main group, if black plays correctly. Net answers with the weak move 13 reducing own territory. JaGo attacks with 14, but Net does not save its stone 7, but attacks (and kills) the white group with the text book move 15. The moves up to 21 are all forced, and explain why 15 was the key move. JaGo passes, and Net plays some superfluent and territory decreasing stones (23–31) until its pass ends the game. Net controls a territory of eleven points and has captured ten stones. JaGo is without territory, but has captured two stones and receives the komi of 5.5 yielding a score of 13.5 in favor of Net.

It must be noted that above game is not the rule, as Net playing black achieves only a win rate of 0.1170 against JaGo. Nevertheless, it shows that Net is able to win and to play some fairly sophisticated Go moves.

In Section 4.2.3 promising results of evolution of recurrent ANNs have been presented. Consequently, we also set up an experimental run with 10,000 generations using recurrent networks for cultural coevolution with, apart from the network structure, parameter settings identical to the feed–forward case. The mean fitness of the population stayed above 0.8, despite the growing culture of increasing strength. Master 10 already had a win rate above 0.9 against Random. From then, all younger masters had win rates above 0.2 against Naive, whereas using feed–forward ANNs the first to reach 0.15 was master 36. The rather low performance of the recurrent ANNs versus JaGo can be explained by their opening moves.

The youngest 21 masters open a game by placing a stone at C2 instead of the optimal C3, which makes JaGo hard to beat. However, in the youngest five masters the intention for opening at C3 steadily increases, which makes it likely that a longer run would produce the optimal opening.

4.3.2 Elite Coevolution

In elite coevolution a fixed number of master networks builds the elite, in which a network from the evolving population replaces a master, if the elite network looses all games

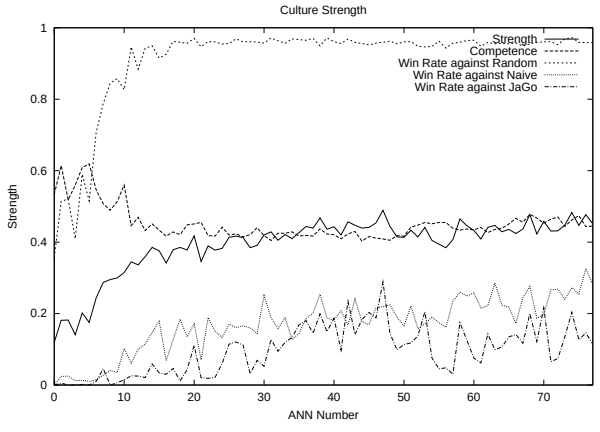

Figure 3: Strength of the master players ordered chronologically (number 0 is oldest of culture).

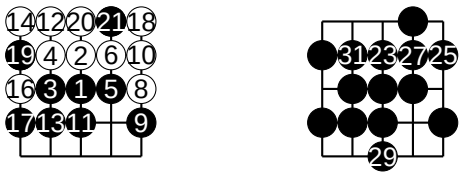

Figure 4: A coevolved ANN (playing the black stones) wins against JaGo (7 at 19, 15 at 21).

against the challenger. With all other parameters being identical to those in Section 4.3.1 we compared elite and cultural coevolution in a run of 3,000 generations with an elite size of 1.

We defined the single elite network after the last generation, and the youngest master of the culture, to be the resulting player of elite and cultural coevolution, respectively. Against our assumptions the elite net was replaced in every single generation, i.e., 3,000 networks have been called into the elite (similar observations have been made with larger elite sizes), which may be an indicator for the occurrence of cycles. The final culture consisted of 49 masters, where the youngest was added in generation 2,705. A detailed comparison of the strength of the two players is given in Table 1.

			Win Rate		
ANN	Strength	Competence	Random	Naive	JaGo
Elite	0.3625	0.4729	0.9230	0.1145	0.0500
Culture	0.4440	0.4052	0.9625	0.2245	0.1450

Table 1: Strengths of neural players generated by elite and cultural coevolution.

Consistently, the stronger culture net defeats each computer player more often than the elite network. This picture did change a bit with increasing elite sizes. In an elite of 16 masters after 3,000 generations three of them exhibited a strength above 40% (0.4012, 0.4403, and 0.4317), however, the strength of all masters in the culture added after generation 500 was above 40%, too.

5 Tournament of Neural Players

Finally, we compare the best networks generated in the various (co)evolution experiments by performing a tournament among them. Each competitor had to play against each other with both, the white, and the black stones.

Four evolved ANNs and six coevolved ANNs entered the contest: The player of greatest strength evolved against Random R, against Naive N, against JaGo (J), and the recurrent net evolved against JaGo (J_r). The coevolved ANNs are the youngest masters of the three culture runs (C_0, C_1, C_2), of the culture run using recurrent networks (C_r), the last elite net in the elite of size 1 (E_1) and the net of greatest strength in the last elite of size 16 (E_{16}). The number of wins of each player is displayed in Table 2.

Interestingly, the two players evolved against JaGo won the fewest games, as these nets learned to pass at the right time, but they are not prepared for players continuing play

	R	N	J	J_r	C_0	C_1	C_2	C_r	E_1	E_{16}
B	5	6	5	2	3	5	4	6	4	5
W	4	3	3	4	5	3	7	6	5	3
Σ	9	9	8	6	8	8	11	*12*	9	8

Table 2: Number of tournament wins of the best (co)evolved ANNs playing black (B) or white (W).

in technically lost games. It can also be seen that C_0 and C_1 having been generated in 3,000 generations could not win more than eight games, but C_2 being the result of 55,000 generations ranked second in the tournament. The tournament winner succeeding in 12 of 18 games is the recurrent culture net C_r produced in 10,000 generations. The performance of the elite nets is similar to C_0 and C_1, as is, surprisingly, the net evolved against Random.

6 Summary

We have presented experiments (co)evolving neural Go players for a 5x5 board utilizing mutli–chromosomal encoding of the players' generalized multi–layer perceptrons. In evolution experiments each of three dedicated computer players of different quality was used as the single opponent of the evolving network population.

In coevolution experiments we introduced a culture representing the Go knowledge of all evolved networks, which receive their fitness according to the win rate against the culture nets. The culture grows dynamically, as a neural player beating all networks in the culture is added to the culture, hence, it must be able to win against players of different quality. For comparisons a different coevolution technique, an elite containing a fixed number of networks, has been implemented, which we expected to exhibit some of the known pitfalls of coevolution. However, in a tournament of the best (co)evolved players the elite and culture networks performed at a similar level. The strength of neural players has been evaluated by the combined win percentage against the three computer players. Though, the strength value of evolved players was greater than those of coevolved players, which never faced human expertise, the first and second place in the tournament went to networks coevolved in a culture.

We also presented promising first experiments with neural players based on recurrent ANNs whose structure is able to reflect neighborhood relations of board intersections, which is not possible with feed–forward networks. In this paper we evolved the structure of the input layer (and all others), which could have connections between any of its neurons representing board interesections, but in future work we will experiment with fixed input layers, where only neighboring neurons (intersections) are connected. Currently, we are working on employing temporal difference learning for the neural Go players, and the extension of evolutionary and reinforcement methods to 9x9 Go boards.

Finally, we would like to thank two of three anonymous referess for their time and effort to provide very detailed, extensive, and constructive remarks.

Bibliography

[1] Claude E. Shannon. Programming a computer for playing chess. *Philosophical Magazine*, 41:256–275, March 1950.

[2] Cho Chikun. *Go: A Complete Introduction to the Game*. Kiseido Publishing Company, 1997.

[3] David Moriarty and Risto Miikkulainen. Discovering Complex Othello Strategies Through Evolutionary Neural Networks. *Connection Science*, 7(3–4):195–209, 1995.

[4] Kumar Chellapilla and David B. Fogel. Evolving an Expert Checkers Playing Program without Using Human Expertise. *IEEE Transactions on Evolutionary Computation*, 5(4):422–428, 2001.

[5] Gerald Tesauro. Temporal Difference Learning and TD–Gammon. *Communications of the ACM*, 38(3):58–68, March 1995.

[6] Sebastian Thrun. Learning To Play the Game of Chess. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 249–252, Cambridge, MA, 1995. MIT Press.

[7] Norman Richards, David Moriarty, Paul McQuesten, and Risto Miikkulainen. Evolving Neural Networks to Play Go. In *Proceedings of the 7th International Conference on Genetic Algorithms*, 1997.

[8] Helmut A. Mayer and Markus Spitzlinger. Multi–Chromosomal Representations and Chromosome Shuffling in Evolutionary Algorithms. In *2003 Congress on Evolutionary Computation*, pages 1145–1149. IEEE, December 2003.

[9] H.-P. Schwefel. *Evolution and Optimum Seeking*. Sixth–Generation Computer Technology Series. Wiley, New York, 1995.

[10] Christopher D. Rosin and Richard K. Belew. New Methods for Competitive Coevolution. *Evolutionary Computation*, 5(1):1–29, 2000.

[11] Robert G. Reynolds and William Sverdlik. Problem Solving Using Cultural Algorithms. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 645–650. IEEE, 1994.

[12] Alex Lubberts and Risto Miikkulainen. Co–evolving a go–playing neural network. In *Coevolution: Turning Adaptive Algorithms upon Themselves*, pages 14–19, San Francisco, California, USA, 7 2001.

[13] Erik C. D. van der Werf, H. Jaap van den Herik, and Jos W. H. M. Uiterwijk. Solving Go on Small Boards. *International Computer Games Association Journal*, 26(2):92–107, 2003.