Salzburg Interest Group on Integrated Systems

siGis-28-2005 June 3, 2005

Evolution of Neural Go Players

Submission to
ÖGAI Journal 2005

Helmut A. Mayer
helmut@cosy.sbg.ac.at

Peter Maier
pmaier@cosy.sbg.ac.at

Department of Scientific Computing
University of Salzburg

Correspondence to:

Helmut Mayer
Universität Salzburg
Fachbereich Scientific Computing
Jakob–Haringer–Straße 2
A–5020 Salzburg
AUSTRIA

Telephone: +43-662-8044-6315
FAX: +43-662-8044-172

Evolution of Neural Go Players

Helmut A. Mayer and Peter Maier
Department of Scientific Computing
University of Salzburg, A-5020 Salzburg
Email: helmut@cosy.sbg.ac.at

Abstract

We present experiments evolving computer Go players based on artificial neural networks (ANNs) for a 5x5 board, where 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 computer Go program from a set of three artificial players of different quality. The playing quality of the evolved players is measured by a strength value derived from games against the three computer players. We also report on results of first experiments employing recurrent networks, which allow a direct structural representation of the Go board.

1 Introduction

With the advent of the first computers board games have attracted many researchers, e.g., (Shannon, 1950), 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* [1], a chess playing IBM supercomputer, defeated Garry Kasparov, the reigning world champion in chess).

The board game Go has received increasing attention in recent years, as unlike chess programs the best Go playing computers are still at a mediocre amateur level, i.e., a good amateur Go player easily beats the machine. 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 like chess or backgammon with about twenty 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). Moreover, in order to win the game not the absolute number of points is decisive but the difference in points the two players have accumulated. In a sense each single game of Go defines itself, as the total number of points varies and depends on the quality and style of the players. 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 believed to have some potential to play Go.

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, todays 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

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

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 (Moriarty and Miikkulainen, 1995).

Chellapilla and D. B. Fogel (1999) 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 250 generations 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 *Class A* level (fourth best category) and could even achieve a win against a human master player (second best category) (Chellapilla and Fogel, 1999).

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 (Tesauro, 1995).

In the evolutionary experiments presented in this work we could generate ANNs beating a Go computer program (based on heuristics) of moderate strength. However, it must be noted that we evolved neural players for the small 5×5 board, while the real game is played on a 19×19 board. The reason for using the small board is the considerable computational cost associated with the evolutionary process, which from todays point of view does not allow board sizes larger than 9×9 (also used by good players for practice games).

2 The Game of Go

The rules of Go (Chikun, 1997) are easy to learn, but the seemingly simple concepts build into deep and complex structures on the board. Two players (black and white) place their stones alternately on the intersections of a 19×19 grid on the initially empty board. Once a stone is played, it may not be moved unless it is captured and taken off the board. The objective of the players is to secure territories by completely surrounding vacant areas on the board with their own stones. A player may elect to pass, i.e., he does not play a stone, and the opponent continues play. If both players pass consecutively, the game is finished.

Stones are said to be adjacent, if they are positioned on horizontally or vertically neighboring intersections. Adjacent stones build a *Group*, e.g., black 1 in Figure 1 does not belong to the two–stone group, but black 2 connects the group with black 1 to form a group of four stones.

A *Liberty* is an empty interesection adjacent to a group of stones. The black group in the left Figure 2 has four liberties, while the black group in the right Figure 2 only has a single liberty.

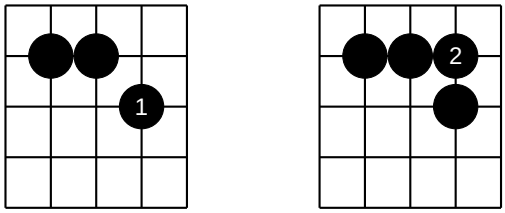

Figure 1: Two groups of stones (left) and a group of four stones (right).

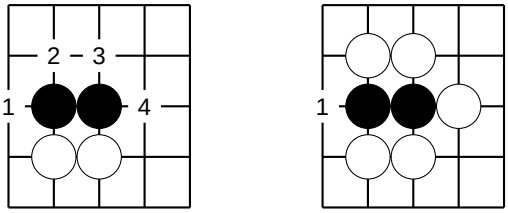

Figure 2: Black groups with four liberties (left) and one liberty (right).

Any group without liberties is said to be dead, and the stones in that group are captured by the opponent removing the dead stones from the board. When playing a white stone at 1 in the right Figure 2 the group of two black stones is dead. However, in most cases white would never play at 1, because black cannot save her stones, and the black stones would be declared dead and be removed at the end of the game. The black group in the right Figure 2 is said to be in *atari* meaning that the group has only a single liberty left.

With two exceptions stones can be placed on any empty intersection on the board. A player may not commit suicide by occupying the last remaining liberty of an own group, e.g., in the left Figure 3 white must not play at 1.

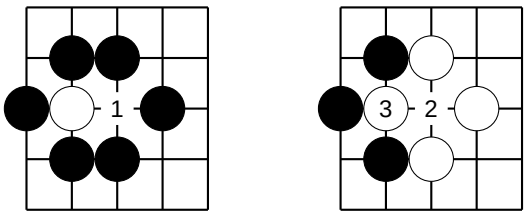

Figure 3: Suicide moves and the ko rule.

However, there is an exception to the exception, namely, a player may (seemingly) commit suicide, if he captures an opponent's group with this move as illustrated in the right Figure 3 by black playing at 2. If black captures white 3, white could immediately recapture black 2 by playing at 3, but the *ko* rule forbids this move. The ko rule prohibits that the same position (on the whole board) reoccurs after two half moves. Hence, it prevents endless repetition and a draw comparable to a *Permanent Check* being legal in chess. Note that white is allowed to recapture black 2 after playing a move elsewhere (*tenuki*). If the tenuki move forces black to react elsewhere, black has no time to secure his stone by playing at 3. This is a key tactical element of Go called *ko threat*.

At the end of the game (both players have passed) the dead stones are removed from the board and added to the opponent's captured stones. It should be mentioned that a correct declaration of the status of the stones (dead or alive) is in some cases a non–trivial task requiring experienced players. Thus, it is even more difficult for a Go program. If human players disagree on the status they may continue play to determine the status, hence, Go programs often leave the status declaration to the human opponent.

After all dead stones have been removed, the score is calculated by counting the number of intersections a player has secured in her territories, and adding the number of captured opponent stones (Japanese score). The white player receives extra points (*komi*) as a compensation for black's advantage to make the first move of the game. Komi values vary, but often a value of 5.5 (the half point prevents draws) is added to white's points. The player with the higher number of points wins, and the final score is the difference of the players' points.

2.1 Eye Shapes and Living Groups

Many different shapes formed by the white and black stones (and the empty intersections) have been categorized and analyzed during the many centuries of Go playing. Simpler shapes restricted to small areas of the board can be recognized easily and the suggested best moves are more or less evident. With growing complexity of the shapes the best moves are often intuitive and finding these moves is the sign of a master player.

One of the most fundamental shapes is an *Eye*, in its simplest form a liberty surrounded by a group (Figure 4).

The black group with a single eye could be captured by white by occupying all outside liberties with atari and then playing at 1. The white group, however, is said to be unconditionally alive, since black after occupying all outside liberties could neither play at 1, nor at 2, both being illegal suicide moves. Hence, the white group is a living group, which cannot be captured, and the eye space is secured territory. In this case the white group secures two territory points, but the eye space could also be larger. With the eye shape concept the Go objective of securing territory can be reformulated into building living groups.

As an example of the complex structures groups on the Go board can create a stalemate state of groups (*seki*) is shown in

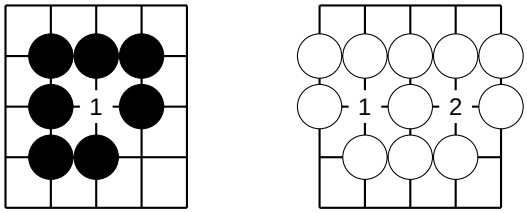

Figure 4: A black group with one eye (left) and a white living group with two eyes (right).

Figure 5.

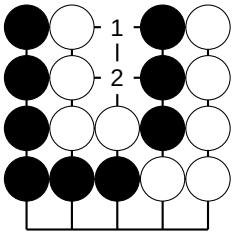

Figure 5: Opposing groups in seki.

Though, at first glance it seems that the player making the next move could easily capture an opponent group, further inspection reveals that for both players a try to attack and capture an opponent group leads only to the loss of an own group. If black played at 1, he would put the left white group in atari, but at the same time his own right group leading to immediate loss of this group (white playing at 2). As the same is true for white playing at 1, both players are well adviced to play tenuki and leave these groups in seki. Seki territory is neutral, because no player has completely surrounded it, and the empty intersection points are not added to the score of either player.

2.2 Rankings and Handicaps

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). As a player learns more about basic Go techniques, he rapidly reaches higher kyu values (from 20 to 15 kyu). This kyu range should be reached after about two months of intense Go studying. Improving the kyu value from 10 kyu up to 1 kyu needs much more time. As soon as an amateur becomes a master *dan* player, she gets the rank of 1 dan (best is 7 dan). Professional ranks are on a scale from 1 to 9 dan. Amateur dan ranks do not correspond to professional dan ranks, as most

professionals are much stronger than any amateur.

The ranking systems reflect the difference of the players' strength, which can be elegantly used to compensate these differences by handicap stones so that both players have roughly the same chances of winning a game. In a handicap game the weaker player always plays black and starts with a number of stones corresponding to the rank difference. These stones are placed on pre–defined intersections (star points) on the board, then, white makes his first move. E.g., when a 2 dan amateur plays a 3 kyu, the 3 kyu receives four handicap stones. The difference of a 1 dan professional to a 1 dan amateur is estimated to be seven handicap stones.

3 Evolution of Neural Go Players

The automatic generation of game playing ANNs by artificial evolution offers some appealing advantages to conventional ANN training. In order to teach an ANN a complex board game the required training data set should ideally cover all aspects of the game including different playing strengths and styles. However, this would lead to an enormous amount of data, which may be impractical for successful 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 (Thrun, 1995) ANNs having been trained with chess games by master players, played reasonably against strong players, but failed to beat weak players. Also, part of the success of ANN training is based on the structure of the network, but no analytical rules to determine the number of (hidden) neurons and the specific connectivities exist, which was a starting point for the field of ANN evolution (Yao, 1999).

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).

3.1 ANN Board Representation

We have extensively experimented with a variety of different board representations for the neural Go player using layered *Feed–Forward* networks. In order to evaluate each representation we trained the networks with a data set acquired from games played by the *JaGo* program against itself, and tested the ANNs by playing games against JaGo and other programs (Section 4). As detection of shapes (patterns) on the Go board is an important quality of a Go player, we have also employed some basic concepts from image processing for the board representation, e.g., *Co–occurrence* matrices. Often, the different representations yielded very similar results, but in the end a simple representation also suggested in related work (Richards et al., 1997) turned out to be the winner.

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.

The output representation simply assigns each output neuron 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 for the 5×5 board (including the pass move).

3.2 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 are shuffled (exchanged) between two parents with a shuffle rate $p_s = 0.5$ (Mayer and Spitzlinger, 2003). 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 maximal 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 chromosome is the standard bit flip mutation, and σ–self–adaption (σ–mutation) (Schwefel, 1995), 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. In order to monitor the development of evolving Go players we devised the following performance measures.

3.3 Performance Measures

We define the strength $s = \frac{w}{g}$ as the win rate of a player having won w games challenging one or more computer Go players in a number of games g. In the following experiments (Section 5) the strength has been measured in games against three computer players (Section 4) 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, in Equation 3 we define 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.

The competence measure is able to separate networks reacting on changing situations in the course of a game from players stubbornly playing the same moves, though, these may be of strategic importance.

4 Computer Go Players

For the 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.

GNU Go [3] is a free Go program being able to play games on 5×5 to 19×19 boards with two to nine handicap stones. It supports two Go protocols, the standard *Go Modem Protocol* (GMP) and the *Go Text Protocol* (GTP) intended to replace GMP in the future, for inter–play with other Go programs.

We used GNU Go 3.2 to determine the strength of JaGo by playing games on a 9×9 board, where GNU Go won about 90% of the games. With the advantage of three handicap stones JaGo won 51.8% of the games, i.e., it is able to play an even game. 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 19x19 board. However, a handicap stone on a 9×9 board is worth more than on a 19 ×19 board. Therefore, JaGo's rank cannot be directly inferred from GNU Go's rating and the number of handicap stones necessary on a 9×9 board. It is assumed that a three stone difference on a 9×9 board approximately corresponds to a difference of six ranks on a 19×19 board, hence, JaGo's rank is about 16 kyu.

Recently, Go on a 5×5 board has been solved (van der Werf et al., 2003). 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

[2] http://www.cs.vu.nl/∼jbmarkes/jago/
[3] http://www.gnu.org/software/gnugo/
[4] http://nngs.cosmic.org/

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.

5 Experiments

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

5.1 Experimental Setup

In all experiments games are conducted on a 5×5 board with a komi of 5.5 for the white player. Evolution (Section 3) 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 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 4).

5.2 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.

5.2.1 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 4). 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.

5.2.2 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. In Figure 6 a typical evolution run against Naive is depicted.

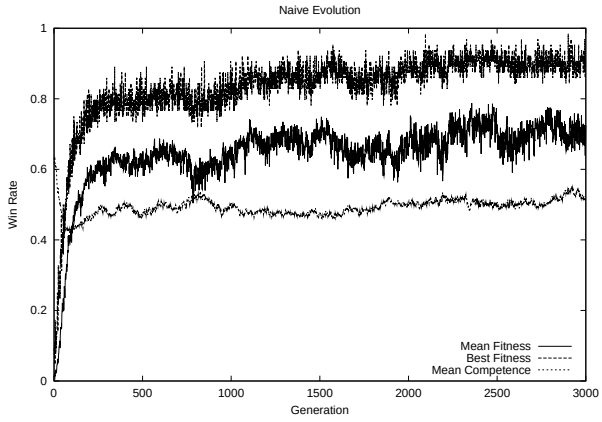

Figure 6: Fitness and competence statistics of evolving players against Naive.

In the last generation the mean win rate of the population is about 0.7. The initially slightly larger competence is a result of nets playing (too) early pass moves, which are always legal. Figure 7 shows the evaluation of the best ANNs (single best of

each run) evolved against Naive with strength values ranging from 0.48 to 0.69.

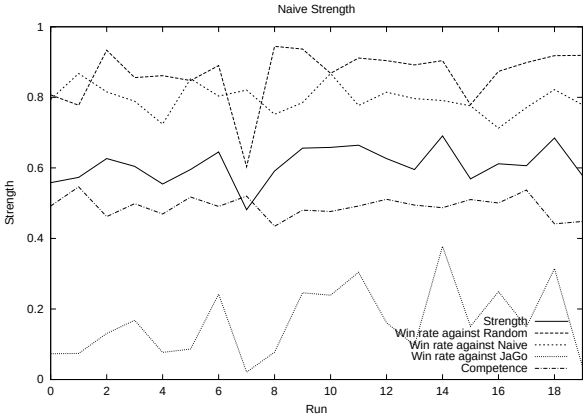

Figure 7: Strength of the best players evolved against Naive.

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. This 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. These ANNs sacrifice their first stone and take advantage of the fact that Naive immediately tries to capture this stone, which gives the net enough time to establish a good position in the center.

In Figure 8 a game between the neural player of lowest strength and Naive is presented, which helps to explain the network's weak performance against other opponents.

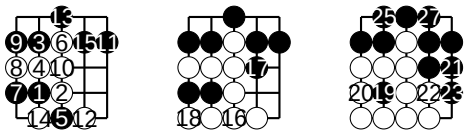

Figure 8: An evolved ANN (playing the black stones) wins against Naive (24, 26: pass).

The net plays a rather weak opening, as after 5 the game would be considered lost. In the mid game the ANN slightly improves by enforcing own stones (7, 9), and attacking the opponent's stones (13, 15). In the end game the net takes advantage of the opponent's mistakes. Naive misses many opportunities to play the key stone 25, which would have captured the upper, left black group and created a white living group. With 25 and 27 the ANN creates a living group at the same time capturing the white group left with a single eye.

5.2.3 Evolution versus JaGo

The next challenger for evolution is JaGo, a fairly sophisticated player (Section 4), 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 9 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.

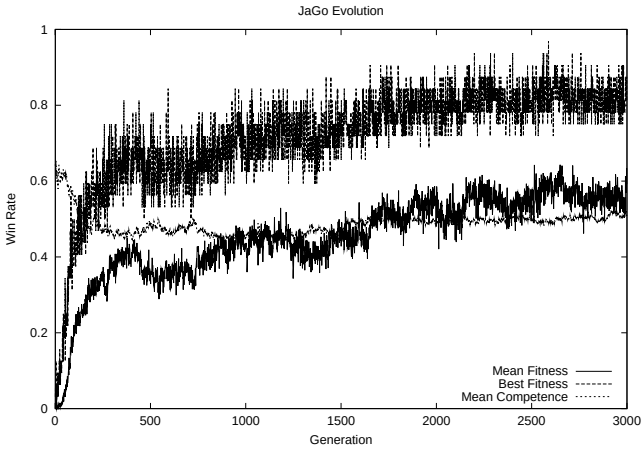

Figure 9: Fitness and competence statistics of evolving players against Jago.

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 and Naive, the neural players often exhibit a preference to place their stones onto key intersections regardless of their state.

Figure 10 illustrates the strength of the best ANNs generated by evolution against JaGo.

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

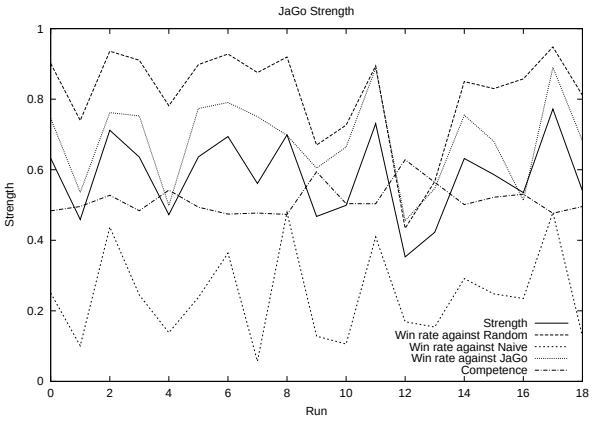

Figure 10: Strength of the best players evolved against Jago.

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 typical game between JaGo (black) and an evolved ANN (strength 0.6360) is displayed in Figure11.

Figure 11: An evolved ANN (playing the white stones) wins against JaGo (18, 22, 24: pass).

Net wins with the smallest possible margin of 0.5 being an indicator for a good quality of play. JaGo has three prisoners and four points of territory, while Net has two territory points and the komi of 5.5. It is most interesting that the dead stone 20 wins the game. If Net had not played this stone, JaGo would neither have placed 21, nor 25, and Net would have lost by 0.5 (two prisoners, and six territory points of JaGo). In this case Net exploits the occasional weakness of JaGo capturing dead stones, hereby, reducing own territory.

5.2.4 Evolution of recurrent ANNs versus JaGo

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 5.1) gives evolution the possibility to transfer board structure to network structure. Due to time constraints we performed only two runs, where each evolved recurrent net played 32 games against JaGo.

Figure 12 shows the evolutionary progress of recurrent ANNs against JaGo.

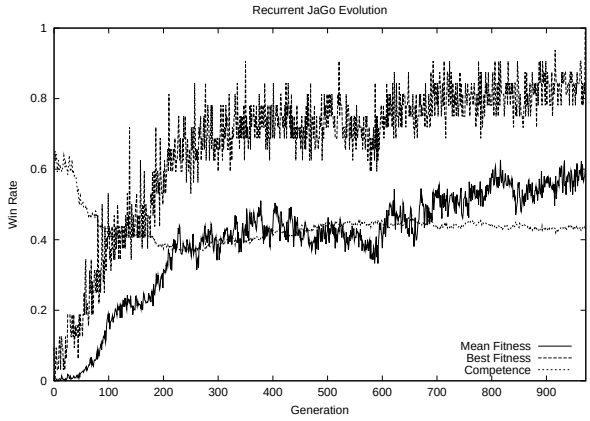

Figure 12: Fitness and competence statistics of evolving recurrent ANNs against Jago.

Compared to evolution of feed–forward ANNs (Figure 9) 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 net from the run in Figure 12 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. Figure 13 presents a characteristic game.

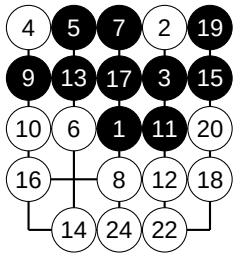

Figure 13: An evolved recurrent ANN (playing the white stones) wins against JaGo.

JaGo opens the game optimally, Net answers weakly, and JaGo attacks 2 with 3. Net responds poorly, JaGo keeps attacking Net by playing 5, and threatens to capture 2 by means of 7. Net ignores JaGo's move and tries to enlarge its territory with 8. JaGo captures 4 with 9. Net's next move is a rather strong move, because it limits JaGo's territory on the upper side of the board. JaGo tries to enlarge its territory, Net defends with 12. JaGo connects some of its stones with 13, which also strengthens its territory. Net prepares for a living group on the lower left of the board with 14. JaGo creates a living group. Net creates two eyes, JaGo gets nervous and puts 17 into its living group to the expense of a liberty and a point of score. Now, Net turns the lower half of the board into own territory. JaGo captures the dead white stone 2 with 19 costing another point of score. 20 enlarges the influence of Net. JaGo passes three times in a row. Inbetween Net plays two stones (22 and 24) reducing its territory. Finally, JaGo owns two prisoners and two intersections, while Net has no prisoners, but it controls three intersections, and receives the komi of 5.5 making Net the winner by 4.5.

This game shows that Net is able to defend own territory, and to create a living group by sacrificing single stones, but it has problems to find the right time to pass. However, this can most certainly be attributed to the facts that JaGo, also, is not always able to detect this moment, and that the score does not add to fitness.

6 Summary and Outlook

We have presented experiments evolving neural Go players for a 5×5 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. Though, evolution always generated networks beating the specific opponent consistently, the known problem of specialization could be observed. E.g., neural players evolved against JaGo, the strongest player utilized in this work, on average beat JaGo in 68% of games played, but won only 25% against the much weaker Naive, which the networks never faced during evolution.

As a consequence, we also have investigated coevolutionary approaches, where neural players only play against themselves without ever being subjected to programs implementing human expertise. 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 (Rosin and Belew, 2000) 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. First results of coevolutionary experiments showed that the neural Go players have a lower strength value (0.44), but exhibit a more general style of play allowing them to win games against all of above computer players as well as the evolved networks presented in this work.

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 hardly possible with feed–forward networks. In this paper we evolved the structure of the input layer, 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 9×9 Go boards.

References

Chellapilla, K. and Fogel, D. B. (1999). Evolving Neural Networks to Play Checkers without Expert Knowledge. *IEEE Transactions on Neural Networks*, 10(6):1382–1391.

Chikun, C. (1997). *Go: A Complete Introduction to the Game*. Kiseido Publishing Company.

Mayer, H. A. and Spitzlinger, M. (2003). Multi–Chromosomal Representations and Chromosome Shuffling in Evolutionary Algorithms. In *2003 Congress on Evolutionary Computation*, pages 1145–1149. IEEE.

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

Richards, N., Moriarty, D., McQuesten, P., and Miikkulainen, R. (1997). Evolving Neural Networks to Play Go. In *Proceedings of the 7th International Conference on Genetic Algorithms*.

Rosin, C. D. and Belew, R. K. (2000). New Methods for Competitive Coevolution. *Evolutionary Computation*, 5(1):1–29.

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

Shannon, C. E. (1950). Programming a computer for playing chess. *Philosophical Magazine*, 41:256–275.

Tesauro, G. (1995). Temporal Difference Learning and TD–Gammon. *Communications of the ACM*, 38(3):58–68.

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

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

Yao, X. (1999). Evolving Artificial Neural Networks. *Proceedings of the IEEE*, 87(9):1423–1447.