

# Evolving poker players using NeuroEvolution of Augmenting Topologies

Morten Lyng Jensen<sup>\*†</sup>  
<sup>\*</sup>Department of Computer Science  
University of Aarhus, Denmark  
U032933@daimi.au.dk

## Abstract

Evolving poker players is not a new idea. Many attempts have been made, using different approaches. This article describes an experiment attempting to improve the evolution of neural networks capable of playing the poker variant Texas Hold'em at tables with 5 opponents by a divide and conquer approach combined with introduction of subpopulations to distribute the algorithm to multiple machines. This method tries to challenge the individuals in different ways in order to identify properties of interest, and the results show potential for this method to develop strong players.

## 1 Introduction

When using fixed topologies for neural networks, one face the problem of sacrificing obtained knowledge when improving the strategy, since altering the network may change knowledge obtained in previous generations. Thereby the fixed topology favours the most significant parameters for the strategy, leading to the loss of less significant information. This less significant information might be useful in some decisions, and should therefore be kept within the topology.

The NEAT method combats this problem by making the topology more complex if necessary. Stanley and Miikkulainen (2002) show that applying the NEAT method for evolving neural networks to a competitive simulated robot duel domain leads to more complex networks as evolution progresses, where the complexification elaborates on existing strategies and the evolved strategies are more sophisticated compared to networks with fixed topologies.

Poker is a probabilistic game with imperfect information, where players can achieve good results by exploiting their opponents' weaknesses. The rules are simple, but the game contains aspects of opponent modelling and calculating odds which are hard to master.

In (Noble, 2002) poker players are evolved by using a fixed topology in sparse networks. These sparse networks consist of only 50 connections, and

are evolved by changing weight on connections or changing source or destination of connections. Noble (2002) states that it is necessary to start out by defining some strategies in the population in order for his implementation to evolve strong strategies.

Ravn and Hansen (2005) use NEAT to evolve poker players. Their strategies are based on information about the game that even novice players could deduce from the cards, the table and the opponent's actions. They evolve players playing heads-up poker, which means that the strategy only has to focus on one opponent. Their experiments show that it is not necessary to implement any knowledge about the game in the population.

In this article, the NEAT method will be applied for evolving poker players for the poker variant Texas Hold'em. The players will be seated at tables with 5 opponents, and will use more sophisticated information in their decision-making.

## 2 NEAT

NeuroEvolution of Augmenting Topologies (NEAT) is a technique of evolving neural network solutions with a minimal topology. This is done by starting out with no hidden layer and then adjust weights while adding hidden neurons and further connections. Only if the strategy with the more complex topology is justified, it can remain in the population.

This is done by speciation of the population and protection of the offspring, so it is not removed prematurely. Furthermore this approach doesn't require any predefined topology, which often is based on experience and experiments.

In NEAT, the population is divided into species based on topological similarities. The distance between two networks is measured as a simple linear combination of the number of excess ( $E$ ) and disjoint ( $D$ ) genes, as well as the average weight differences of matching genes ( $W$ ).

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 * W$$

Figure 1: The linear combination used to determine the distance between 2 individuals

The coefficients  $c_1$ ,  $c_2$  and  $c_3$  are used to adjust the importance of the 3 factors and the factor  $N$ , the number of genes in the larger genome, normalizes for genome size. If the distance ( $\delta$ ) between a genome and a randomly chosen member of a species is less than the compatibility threshold, the genome is placed into this species. The values in table 1 are used during the experiment and are found through trial and error, based on the demand of having approximately 15-20 species in each population leading to an average species size of 6.

Table 1: The values used in experiments

Compatibility threshold	1.5
$c_1$	0.6
$c_2$	0.6
$c_3$	0.5

NEAT protects offspring by using *explicit fitness sharing*, which means that a given species' fitness is calculated from its genomes fitness. This must be done because an offspring's fitness cannot be expected to be as good as its parents' fitness, and therefore the offspring could be removed from the population prematurely. Furthermore *explicit fitness sharing* ensures that a given species cannot afford to become too huge, since a species contributes with new offspring based on its shared fitness. This prevents one species from taking over the entire population.

Stanley and Miikkulainen (2002) show that NEAT is performing well at solving the *double pole balancing with velocity information* (DPV) problem, while finding more minimal solutions than other techniques. The principle of starting out with no topology however goes against starting with some pre-

defined knowledge as done in (Noble, 2002). In the experiment presented in this article, some simple knowledge was implemented in the start population by adding connections from some of the input neurons to the output neurons.

### 3 Method

Texas Hold'em consists of two stages, the preflop and the postflop<sup>1</sup>. Therefore the use of two distinct networks, each covering one of these parts of the problem domain is initiated. This is done, because the evolving of a single network handling both stages is assumed to favour the first stage. If a strategy decides to fold during preflop, the postflop part of the strategy is not utilized, hence, the information from the postflop part to the genetic algorithm will be limited. Furthermore 5 networks are used to handle the opponent modelling. Each of those handles a specific number of opponents, so if the player is facing 3 opponents, the network handling 3 opponents is used. First the preflop or postflop network decides which action to perform. This action is then fed to the proper opponent modelling network, which – if needed - modifies this action accordingly to the opponents' way of playing.

The preflop network consists of 34 input neurons, the postflop network consists of 78 input neurons and the opponent modelling networks consist of 55, 70, 85, 100 and 115 input neurons respectively. All networks have 3 output neurons, fold, check/call and bet/raise. Depending on the stage of the current game, information is fed to the proper network, and the network is activated. The decision of the network is then decided by the output neuron with the highest activity level.

The information available to the network is either Boolean or continuous. The continuous are squeezed with the function  $y = \frac{x}{1+|x|}$ , in order

to be in the interval ]-1;1[. At first, the input was normalized using the function  $y = x * 2 - 1$ , but since the number of inputs set during decision is outnumbered by the number of inputs not set, this normalization blurred the inputs set, since all not set inputs was normalized to -1. This caused the strategies to take unset inputs into consideration, which is why the inputs are not normalized. The information tries to cover all aspects of the game, from information about the hole cards held by the player, to information on the pot odds and how many better hands can be made from the community

<sup>1</sup> Knowledge of the rules of Texas Hold'em is assumed

cards, to actual opponent modelling, where the opponents are considered to be either loose/aggressive, tight/aggressive, loose/passive or tight/passive. Loose/tight refers to the number of hands played by the opponent. The opponent is loose if he participates in many hands and tight if he carefully selects those hands, he's playing. Passive/aggressive refers to the number of raises. If an opponent raises, given the option, most of the times, he's said to be aggressive, whereas an opponent just calling in those situations is passive.

To improve the decision making, information on all, latter half and latter 10% of the hands played by each opponent are fed to the proper opponent modelling network.

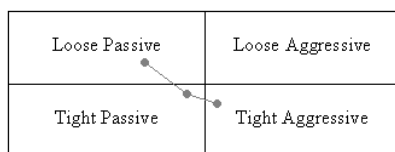


Figure 2: An opponent's table image. The dot to the left indicates all hands seen, the middle dot indicates the latter half and the dot to the right indicates the latter 10% of hands seen. The opponent is changing his style of play. Generally he's playing loose/passive, but recently he's been playing tight/aggressive.

Figure 2 shows that a player can change his playing style over time and that, if the opponent modelling only keeps track of the entire set of hands seen, opponents can change playing style without any major changes in the general playing style.

Information on the opponents is maintained during the game, so as time goes, more accurate information is available. The amount of information is too complicated for most human players to handle at once, but hopefully the results will indicate, which information is most important when playing poker.

The population sizes used during evolution was 108 for all strategies. These populations were kept at a so-called Delegator, which handles the distribution of subpopulations and the genetic algorithm. In each generation step, each population is divided into subpopulations, which is sent to different environments to be evaluated. These environments challenge strategies in different ways. 5 environment types are used; merged, preflop, postflop, model and benchmark. In the merged environment, 18 complete players are evaluated through games against each other at 3 tables. The preflop, postflop and model environments evaluate the preflop, postflop and model part of a player respectively each taking a subpopulation of 36 strategies. The benchmark environment indicates improvements in the populations through games against 3 hand coded strategies;

a tight, a loose and a tightbluffer. 4 merged, 1 preflop, 1 postflop, 1 of each model and 3 benchmark environments; one inhabited by tight players, one inhabited by loose players and one inhabited by tightbluff players was initiated during the experiment. The number of environments determines the number of individuals in each population, since each strategy should be evaluated in each generation. Each environment is assured to play at least 600 hands, but instead of waiting for the slower environments to complete their hands, another 120 hands is carried out. Once all environments have played at least 600 hands, the results are returned to the Delegator, where the next generation step is performed.

As in (Noble, 2002) the strategies' winnings were stored in a matrix during the games. A strategy A is said to dominate another strategy B if A's performance against all the strategies in the subpopulation is at least as good as B's performance against all the strategies in the subpopulation and better against at least one. The number of strategies dominated by a given strategy was used to indicate that given strategy's fitness. The 3 most dominant strategies from each merged environment played against a hall of fame of older dominant strategies from previous generations. Only the strategies placing top 3 against all groups of older dominant strategies were added to the hall of fame.

## 4 Results

The experiment was carried out on 15 P4 3 GHz machines and ran for 14 days resulting in 101 generations. During the experiment 9 randomly selected strategies was sent to all 3 benchmark environments to measure which type of players the populations are performing best against. The results showed that more of the strategies performed better against all three types of benchmark players, which confirms that the genetic algorithm develops robust strategies.

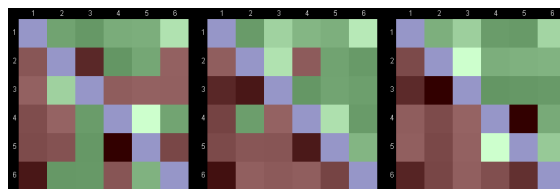


Figure 3: Results from the tight environment. The 3 matrices is from generation 26, 51 and 101. Players from the populations are placed as 1, 2 and 3, and the tight benchmark players are placed as 4, 5 and 6. Blue indicates no development, which only appears in the diagonal. Green indicates improvements from generation 1 and red indicates aggravations from generation 1. The matrices are read by starting at the row for a given player. The colour in a given column

indicates how the row player performed against the column player.

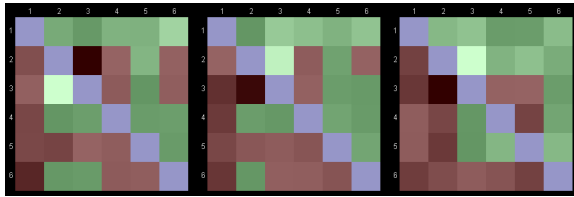


Figure 4: Similar results from the tightbluff environment. Player 1, 2 and 3 are the same as in figure 3.

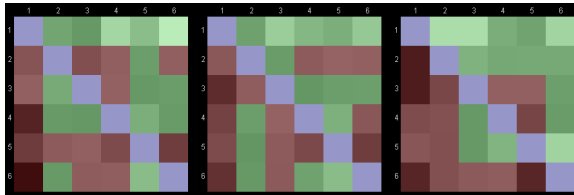


Figure 5: Similar results from the loose environment. Player 1, 2 and 3 are the same as in figure 3 and 4.

Figure 3 shows how 3 randomly selected players from generation 26, 51 and 101 respectively are performing against the tight benchmark players. Compared to the results in figure 4 and 5, one can conclude that player 1 and 2 in generation 101 is robust, since these players perform well against all others opponents in all benchmark environments. In the tight environment all 3 players from the populations are doing better against the benchmark players, which indicates, that the populations abilities to play against tight opponents have increased. It seems however that player 3 in generation 101 is performing worse against the tight bluffers and loose players. This suggests that this player is specialized against tight players.

By examining how often new dominants are found, it can be stated how fast the algorithm finds stronger strategies.

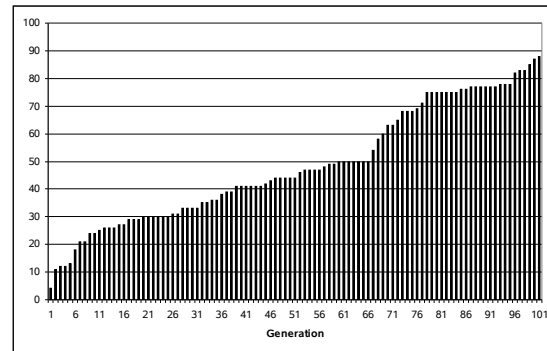


Figure 6: How many dominants are present in each generation.

Figure 6 shows that the number of dominants increases in clusters. This behaviour was also observed in Ravn and Hansen (2005). This suggests that the dominants occur once the algorithm finds an augmenting topology with reasonable weight adjustments and that this solution is improved during the following generations, leading to more dominants. After a while, the algorithm cannot improve the solutions any further without altering the topology, which needs to be adjusted before new dominants are found.

This is also supported by looking at the average number of hidden neurons in the populations and the dominants.

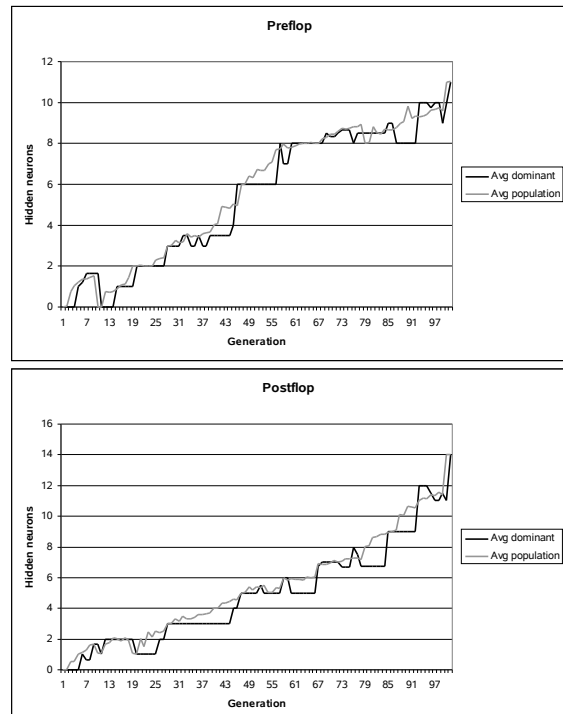


Figure 7: The average number of hidden neurons in the dominant strategies and in the strategies in the preflop and postflop populations.

Figure 7 shows that the dominants have a lesser

complex topology than the populations in general. Once a dominant is found, the number of hidden neurons in the new dominants is not increased before the more complex topology is properly adjusted.

These 101 generations are enough to show that the distributed algorithm finds more complex solutions and that the future solutions are stronger. The stronger strategies utilize more advanced knowledge than introduced in the start strategies. This motivates for further generations to be commenced.

## 5 Future work

This approach can easily be extended to incorporate more environments and it would be interesting to examine the efficiency of introducing more environments. Some environments are likely to contribute more to the evolution, and locating those would also be an interesting aspect to cover. Currently the individuals are evaluated against each other, which require the population to offer the necessary resistance for the individuals to evolve. For instance implementing some of the rules described by Sklansky and Malmuth (1999) and use those to challenge the population would provide increased learning rates.

Another interesting thing would be to replace the NEAT technique with a real-time NEAT technique. This has been used to improve performance for agents interacting in hostile environments by Stanley, Bryant and Miikkulainen (2005). By replacing a poor performing individual with an offspring created from 2 well performing individuals every once in a while, the population evolves during the game in real-time. Human interaction contributes to the abilities of the population, and one can take the role as a trainer and construct different environments in order to stimulate the population during evolution.

## Acknowledgements

Special thanks are due to Brian Mayoh for his interesting points of view and feedback.

## References

- Kenneth O. Stanley and Risto Miikkulainen. Continual Coevolution through Complexification. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*. San Francisco, CA: Morgan Kaufmann, 2002.
- Kenneth O. Stanley and Risto Miikkulainen. Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation 10(2)*. MIT Press, 2002.
- Jason Noble. Finding robust Texas Hold'em poker strategies using Pareto coevolution and deterministic crowding. *Proceedings of the 2002 International Conference on Machine Learning and Applications (ICMLA'02)*, CSREA Press, 2002.
- Mathias Ravn and Lars Vadstrup Hansen. Neuroevolution of artificial poker players without pre-encoded expert strategy. *Master thesis in Danish*, Daimi, 2005.
- David Sklansky and Mason Malmuth. Hold'em Poker for Advanced Players: 21<sup>st</sup> century edition. Two Plus Two Publishing, 1999.
- Kenneth O. Stanley, Bobby D. Bryant and Risto Miikkulainen. Real-Time Neuroevolution in the NERO Video Game. *IEEE Transactions on Evolutionary Computation (Special Issue on Evolutionary Computation and Games)*. Vol. 9, No. 6, December 2005.