EVOLVING AND COEVOLVING COMPUTER GO PLAYERS USING NEUROEVOLUTION

Wester Zela Moraya **Jose Zato Recellado**
edzela@yahoo.com zato.upm@gmail.com
Facultad de Informática
Universidad Politécnica de Madrid

ABSTRACT
This work reviews some neuroevolution techniques used in reinforcement learning applied to the GO game. Go is ancient very complex game with simple rules which still is a challenge for the AI. This work is reviewing the SANE (Symbiotic Adaptive Neuro-Evolution) method and presenting a variation with the intention of evolving better strategies in the game. It is proposed the co-evolution as a solution to the problem of deterministic players, players able only beat with which were trained. Finally, it is introduced an algorithm to co-evolve two populations of neurons to evolve better Go players.

KEY WORDS
Go, Evolution, Coevolution, Neuroevolution, SANE, SANEi.

INTRODUCTION

The GO game is an old game that has started in China thousand years ago, and his popularity has been grown around the world in the last years, nowdays there are many tournaments around in Europe, USA and other continents (i.e. European Go Tournament [1]).

The official game is played with two players using white and black stones in a board of 19x19 lines. For training propose is used boards of 9x9 lines and 13x13 lines. The main object of the game is to use stones to surround a larger portion of the board than the opponent. Once the stones are placed on the board, this cannot be moved, except in the case that they are captured by the opponent. When a game finishes, the controlled intersection or territory, is counted along with captured stones to determine what the scores of the players is. The player has the possibility to pass his turn, when three passes are executed continuosly the game end and the score is calculated. There are two general strategies in the game, one is placing stones close together usually helps them support each other and avoid capture, and second is placing stones far apart to creates influence across the board Part of the strategic difficulty of the game is finding a balance between these types of strategies.

The basic principle is that stones should have liberties (be next to empty intersections) to remain on the board. A liberty is an empty intersection next to a stone. The main objective of the Go player is to expand the one’s where possible and attack the opponent's weak groups (groups which can possibly be killed), and always stay mindful of the “life status” of one's own groups.

The Go game still is a challenge for the Artificial Intelligence because has few rules, but at the same time is very complex because of the number of movements and strategies that can be applied in the board. In the last years were explored different techniques to create good computer Go players as described by Bouzy and Cazenave [4] or by Bernd Brügmann [16]. Some of these computer Go players have won some games against professional players as many faces of Go, MoGo, Crazy Stone in boards of 9x9 or 13x13 lines with handicaps of 6 or 9 stones [2].

MoGo is a computer player who is based in Monte Carlo and Tree Search (TS) algorithm (which is originally based in the UCT algorithm) [3] and on August 7, 2008, this computer program running on 25 nodes (800 cores, 4 cores per node with each core running at 4.7 GHz to produce 15 Teraflops) of the Huygens cluster in Amsterdam beat to professional Go player Myungwan Kim (8p) in a nine stone handicap game on the 19x19 board on the KGS Go Server [3].

The contribution of this paper is the revision of some Neuroevolution (NE) techniques and co-evolutionary algorithms applied to computer Go. These techniques have shown good results in complex reinforcement learning task as applied by Gomez and Miikkulainen [9]. The advantage of NE applied to computer Go is to train some Go players without any previous knowledge of the game, saving effort of writing the strategies of the game in the program as other computer Go programs. The results obtained using NE are good in the experiments executed. It is proposed a variation of SANE to train computer Go players with better results. After the training against a known player was observed that the computer Go players generated are able only to beat players against they were trained, creating deterministic players, to solve this problem were introduced the competitive co-evolution to evolve two different populations competing players against each other.

REVISION OF SOME NEUROEVOLUTION (NE) TECHNIQUES

There are different NE techniques for evolving weights and weights and the structures of neural networks, the ones reviewed in this work are the following: SANE (Symbiotic Adaptive Neuro-Evolution), ESP (Enforced Sub-Population), NEAT (Neuroevolution of Augmenting Topologies).

ESP (Enforced Sub-Population)

The main feature of this technique ESP applied to Go is the definition of some regions (i.e. 3x3 lines) as different populations which are evolved as separate population, getting diversity and specializing some neurons to some specific problems or to specific regions (i.e. play in the corner). According to Perez-Bergquist [5] in general ESP has been more effective that SANE because with the same conditions in the experiment SANE needed networks of 300 neurons to defeat the Gnugo (a known player), but ESP only needed 10 neurons (2 hidden layers). But, the issues faced using ESP is that can't be scalable to bigger boards, for example, good players that evolved in a board of 7x7 were not possible to be used in 9x9 board, in fact the networks that had better performances in the evolution were the networks that started from scratch and not moved from one small board to big board [5].

NEAT (Neuroevolution of Augmenting Topologies)

NEAT is technique to evolve weights and the structures (topology) for NE proposed by Keneeth [6] which belongs to TWEANN (Topology and Weight Evolving Artificial Neural Networks) techniques. The main benefits are the following:

- Don't lose the time to find manually the best structure of the neuron population
- Can evolve from simple structures to more complex structures as in the nature, in similar way that evolving simple strategies to more complex strategies.
- Protecting the innovation through speciation or niching, using historical markings to identify which genes are coming from the same root or parents. The idea is to divide the population into species to compete into their niches protecting the innovation (new structures from mutation) and compete later in the large population.

- Mating similar species using the historical marking of each gene (using the *differences function*).

According to the author, two genes with the same historical origin must represent the same structure [6], although with different weights, since both are derived from the same ancestral gene at some point in the past. To track this historical origin the author propose a *global innovation number* which is number added to the system every time that a new gene that appears. It was demonstrated using the experimental comparison in the task of pole balancing problem that NEAT is more efficient to others ESP, SANE or CE [10].

SANE (NeuroEvolution of Augmenting Topologies)

SANE is a NE with fix structure where the weights of the structure of the networks are evolved proposed by Moriarty [7] and proved in different problem with good results as [12] , [13]. The following figure show how the neuron and blueprint network are built.

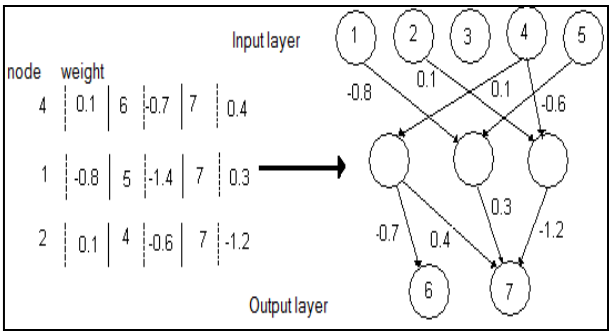


Figure 1. Structure of a Neuron

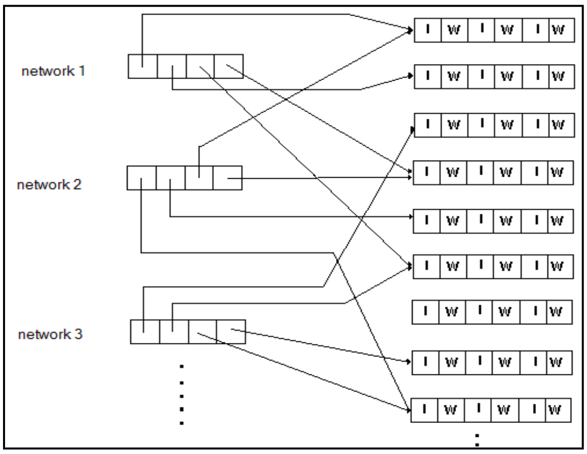


Figure 2. Structure of the blueprint or network

The neuron contains nodes (that could be the input or the output to the hidden layer) and weights that connect the hidden layer with the input/output. The activation of the neuron is calculated between the sum of the all input and output multiplied by their weights and passed through this sigmoid function $\sigma(x) = 1/(1 + e^{-x})$. The size of the neuron (or gene) is the number of nodes and weights in the gene.

The network or blueprint points to a set of neurons of the population of neurons in every generation. The relation between the neuron population and the blueprints can be observed in the fig. 2. The same neuron can belongs to more than one blueprint in every generation. The offspring of the members of the population is a sexual offspring being the parent the best neurons from the previous generation.

EVOLUTION OF COMPUTER GO PLAYER

SANE maintains the strategies or knowledge of the game keeping the best networks (which have the best fitness) and evolving them through generations. In the same way the neurons that participate in the networks with the best fitness are maintained and evolved in the population. The best neurons are cross replacing neurons the worse and some of the worse are mutated to get new members.

For this work was selected SANE because is easy for implementation and because the main assumption is that all members of the populations to evolve are the same specie (which can be crossed each other). Although other methods as NEAT has demonstrated better results than SANE for some problems, the proposed variation of SANE, called SANEi needs that members of the population belong to the same specie.

SANE

SANE has two parts, the evaluation and the reproduction phase. In the evaluation part, SANE simultaneously evaluate the blueprints networks and the neurons. The network is evaluated by the performance to solve problems; as in this case the problem is to beat to Go player, the best solution to the problem is the network that gets the best score against another Go player. The neurons are evaluated based in the performance of the network in which the neurons are participating. The basic steps in evaluation phase are the following [7]:

```
Per each neuron n in the population Pn (initialization)
        n.fitness ← 0
        n.participation ← 0
Per each blueprint b in the population of Pb
        neuralnet ← decode (b)
        b.fitness ← task (neuralnet)
Per each neuron n in b
        n.fitness ← n.fitness + b.fitness
        n.participation ← n.participation + 1
Per each neuron n in the population Pn
n.fitness ←  n.fitness / n.participation
```

The score of every player (network of neurons) in the game is added to fitness of the each neuron that belongs to the network. After all networks have been evaluated (played against other go players), the fitness of each neuron is normalized by dividing the sum of the scores by the number of total networks in which the neuron has participated.

In the reproduction phase, SANE uses all the genetics operators as crossover and mutation to get new blueprints networks and neurons. In case of crossover, every population (neurons and networks) is ranked based on their fitness and is defined an elite of members in each population which will used for mating to other members of the population replacing the members who the worse performance. In case of mutation, the members of the population with worse fitness are mutated.

As Alex Lubbert, Risto Miikkulainen [11] mentions that evolving neurons instead of complete networks, the search space is decomposed and groups of neurons are able to specialise on different parts of the task. This way, diversity is maintained and the algorithm does not get stuck on a suboptimal solution and the blueprint population then searches for effective combinations of neurons.

SANEi

This work is introducing a variation to SANE method which was called SANEi, because of the introduction of immigrant population in every generation during the evolution in the production part of SANE. SANEi includes an immigration rate to introduce the new neurons in the population of neurons replacing the neurons worse ranked in every generation. This new members in the population are creating a real infinite population of neurons, which apparently is not happening with the genetic operators a crossover or mutation, as is discussed below. The introduction of new neurons in every generation is creating a major diversity, and there are indications that SANEi is creating more strategies in the game. The fig. 3 shows the architecture implemented to be used in this work.

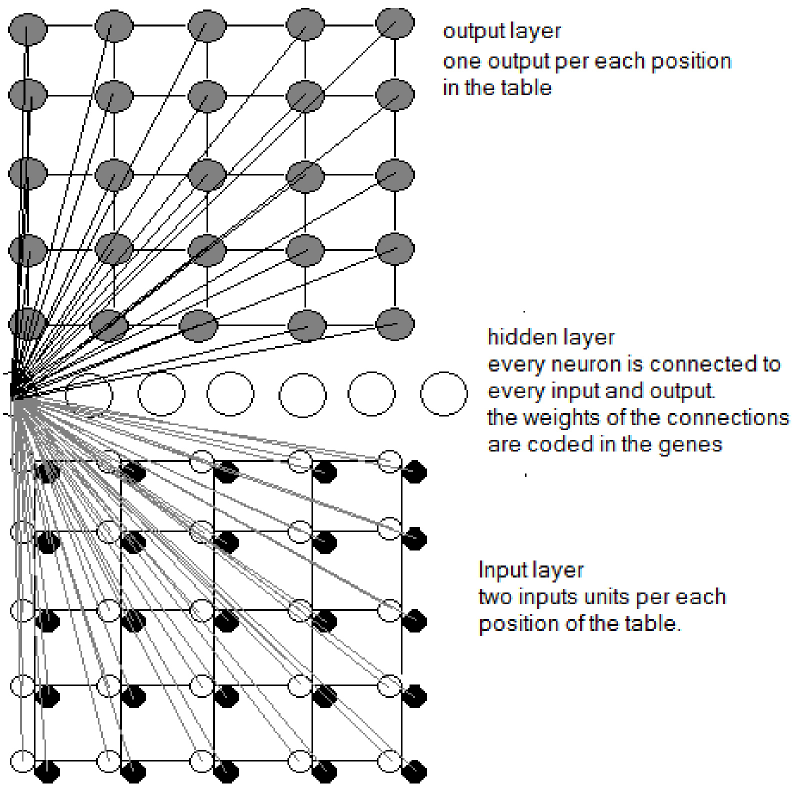
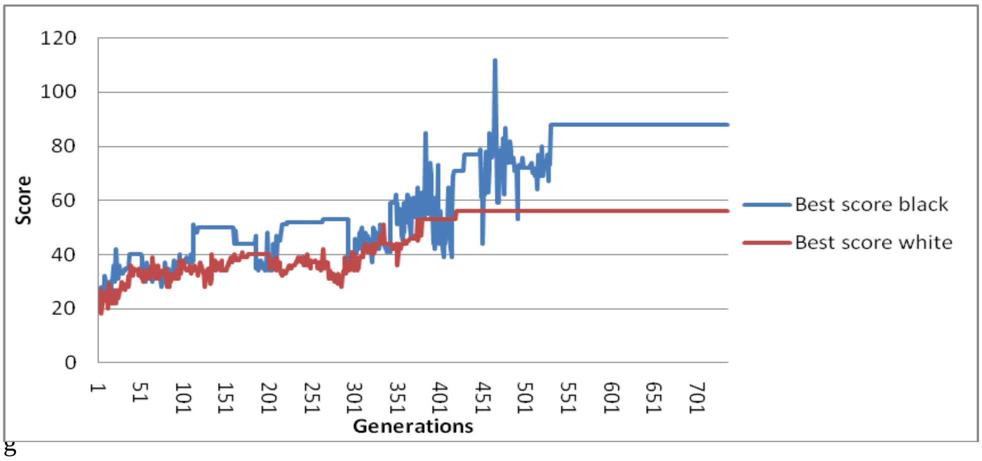


Fig. 3. Architecture of the Neuronal network implemented for SANE and SANEi. For every intersection in the board of 9x9 there are two inputs and one output.
The hidden layer connects the input and the outputs.
For the Input layer was include two more inputs which are the last movements of White and Black stones in the board.

The framework used to run all the executions is the OpenGo which is free available (http://sourceforge.net/projects/opengo)

Results of playing SANE and SANEi against existing player

In the previous paragraphs is described how SANE/SANEi evolves the populations of neurons and networks to find the best strategies to beat the opponents. For training SANE/SANEi was selected a computer Go program called Wally, which is publicly available at (http://www.joerch.org/go/wally.html). This Go program is not stronger enough, but is useful to demonstrate how SANE/SANEi evolves the strategies against this player from scratch. In the following experiments is used a board of 9x9 lines. SANE/SANEi was executed with different number of populations of neurons and networks, but after many executions was identified that the following is the correct configuration for this work. In the fig. 4, SANE is playing with a population of 1000 neurons and networks of 300 neurons. The parameters used are the following, crossover rate 50%, mutation rate 3% and immigration rate 0%. The gene size of the neuron is 216.



g
Fig. 4. Execution of the black and white stones using SANE against Wally in the board 9x9

In the fig. 5, SANEi is playing with a population of 1000 neurons and networks of 300 neurons with parameters as crossover rate 50%, mutation rate 3% and immigration rate of 3%. It is the same configuration used in the previous experiment with the only difference is the immigration rate in SANEi is greater than Zero.

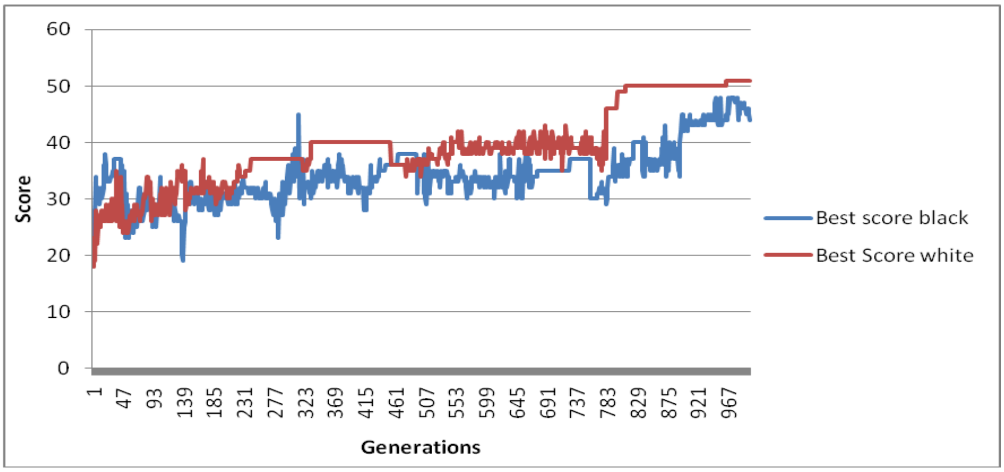


Fig. 5. Execution of the black and white stones using SANEi against Wally in the board 9x9

The intention of these two experiments is compare SANE and SANEi which ones producing better evolution (or getting better fitness) in similar environments. Evolving the same initial populations of networks and neurons for black and white stones, can be observed in fig. 4 and 5, that SANE starts to beat more early to Wally and get better scores than SANEi, but after some time (generation 400 for white stones and 510 generation black stones) is observed that SANE is not getting better fitness (playing white or black), which can indicates that this player is not learning new strategies. This is indicating as well that the genetic operators are not creating diversity in the population after some time. By contrary, SANEi take more time start to beat Wally but continue the learning in the time, because the diversity introduced by the new neurons in the populations. For the execution of these experiments not handicap and Komi was used and the score of the players are calculated using the Japanese scoring.

COMPETITIVE CO-EVOLUTION

The previous paragraphs have demonstrated that the algorithm SANE and SANEi can beat an existing computer GO player as Wally and others as GNUgo as was demonstrated by Lubberts et al [11]. The problem of the strategies generated for these two methods (and others for which evolution is applied) is that they are deterministic; it means that the players evolved can beat only to these existing players with which were trained. This is a major problem if the intention is to use the trained players against other computer Go players or compete against to human professional players which should be ultimate goal.

This problem is addressed in the following way, co-evolving two populations of neurons and networks trying to beat each other in every generation. This way of evolution is called competitive co-evolution as was applied by Rosin and Belew [14] in other problems.

According to Alex Lubberts, Risto Miikkulainen [11], one way to define competitive co-evolution is by evolving two populations: one is a population of hosts that try to find an optimal solution; the other is a population of parasites that instead of trying to find an optimal solution, try to defeat the hosts by making use of their weaknesses, applying asymmetric arms race principle [15] where two different species or populations compete against each other. In this work the host population is playing black stones and the parasite population is playing white stones.

Evolutionary Algorithm for Co-Evolving two Computer Go players

This is the strategy proposed to address the deterministic problem and evolve better strategies for black and white stones:

Training of the populations for co-evolution:

- Train the initial populations of neurons of black (host) and white (parasite) stones players against opponent (Wally or other computer Go player) using SANEi (or SANE).
- When the players are starting to beat the opponent or when the populations are good enough trained with good scores, stop the evolution of these two populations.

Use the populations trained in the previous steps to start the co-evolution of black player against white player using SANEi (or against SANE). The next steps are repeated in every generation:

- In every generation N x M interactions will be produced. Where N, M is the number of networks of neurons per populations.
- A competitive fitness sharing (Rosin and Belew [14]) is used to calculate the fitness of every player in every generation. The intention is to keep for the following generations the sample of hosts (or parasites) that can be only defeat the parasites (or host) that other hosts (parasites) are not able to defeat. The fitness is calculated in the following way:
    - Competitive fitness sharing for the host (black player):

      If host(i)→fitness > parasite(j)→fitness

      host(i)→fitness $= \sum_{i=0}^{i=n}$ (1/Number times parasite(j) lost)

      The number times parasite(j) lost: the number of times that the parasite(j) lost against other parasite players in the same generation.
    - Competitive fitness sharing for the parasite (white player):

      If parasite(i)→fitness > host(j)→fitness

      Parasite(i)→fitness $= \sum_{i=0}^{i=n}$ (1/ Number times host(j) lost)

      The number times host(j) lost: number of times that the host(j) lost against other host players in the same generation.
- The populations of networks are ranked based in the fitness shared obtained in the generation. The populations of neurons are ranked based on the fitness of the networks were the neuron has participated.
- In the production part, the populations of neurons and networks are evolved using crossover, mutation and replacing the worse ranked of the population of neurons with new members based in the immigration rate (SANEi).

Results of Co-evolving two Computer Go players

The experiments has been executed in two ways, co-evolving trained and not trained population of neurons and networks to compare which populations (strategy of co-evolution) obtain better results. For both executions was used the same configuration. Population of 1000 neurons, networks of 300 neurons, crossover rate of 50%, mutation rate of 3%. In every generation were trained 30 networks playing against 30 networks of the opponent (900 interactions or games per generation). For SANEi the immigration rate is 3%, replacing the worse 200 neurons of the population in every generation.

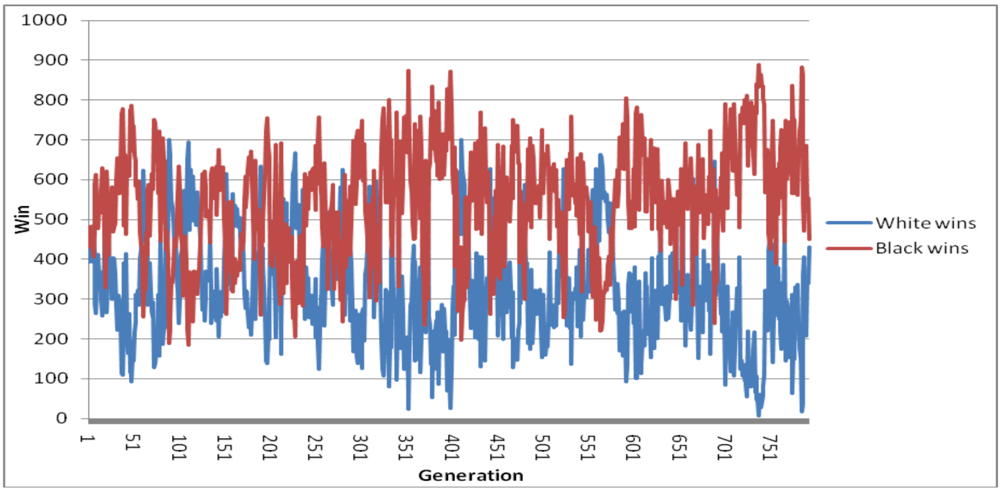


Fig. 6. Co-evolution of SANE playing White stones and SANEi playing Black stones (initial population not trained)

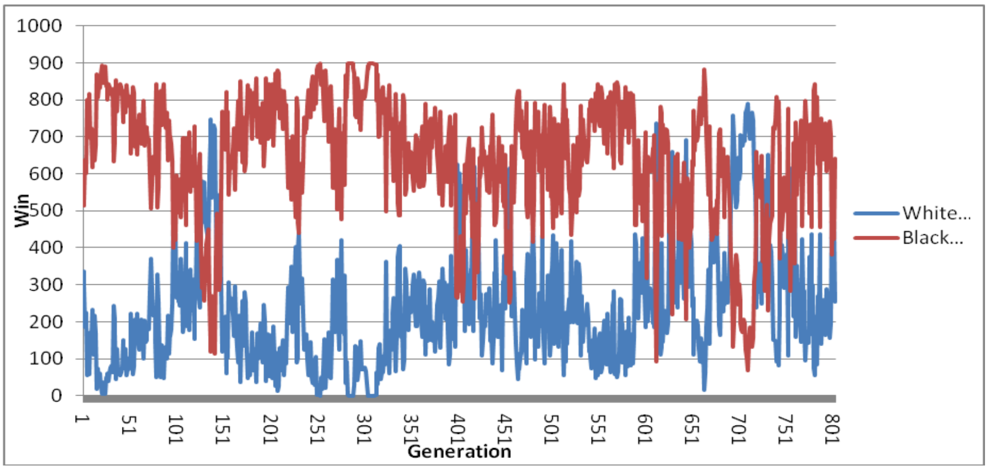


Fig. 7. Co-evolution of SANE playing White stones and SANEi playing Black stones (initial population trained)

In the fig. 6 and 7 is observed the co-evolution of two populations where SANEi is playing black and SANE is playing white. These results are indicating that in both scenarios, co-evolution with trained and not trained initial populations, the players co-evolving with SANEi beat more frequently to the players playing SANE. The same was observed in other executions using different initial populations.

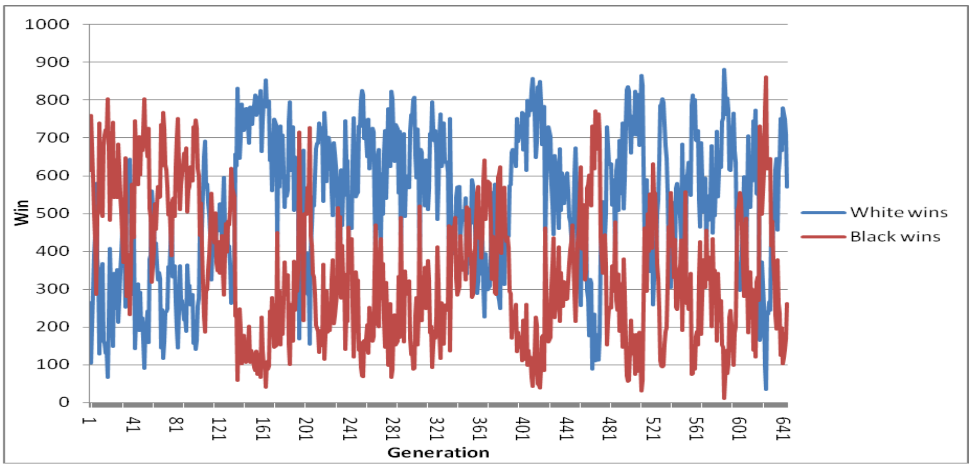


Figure 8. Co-evolution of SANEi playing White stones and SANE playing Black stones (initial population no trained)

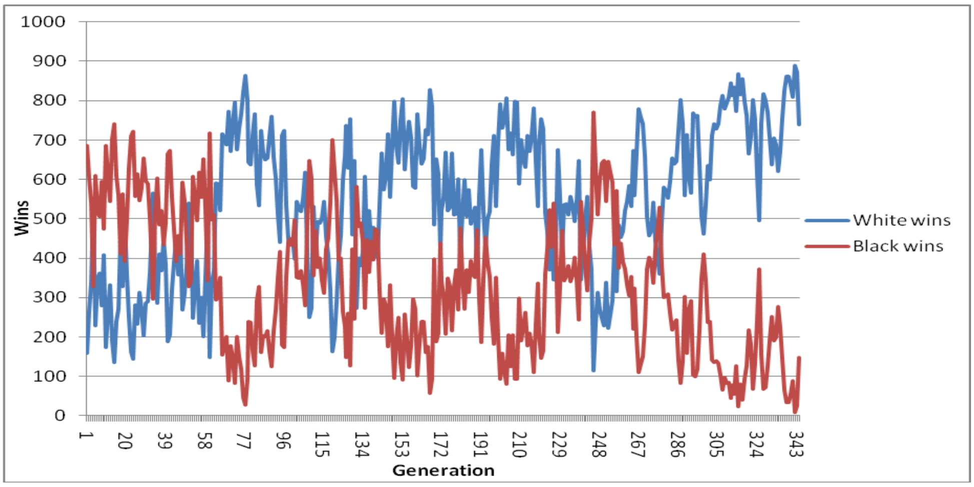


Figure 9. Co-evolution of SANEi playing White stones and SANE playing Black stones (initial population trained)

In the fig. 8 and 9 is observed something similar to the previous co-evolutions, in this case SANEi is playing white and SANE is playing black stones. The previous results indicate that SANEi playing black or white stones in the time have better results than SANE. The populations initially trained in the co-evolution of these populations shows that are beating more frequently to SANE than the populations not trained initially. For a future work the populations should be trained initially against a stronger opponent as GnuGo or others.

New tests were executed with some modifications to the program which incorporated two basic rules:

- Not put the stone in opponent's eye (and avoid the rejection of the move by the referee of the OpenGo)
- Not put the stone in the intersection that has not liberties (have more possibilities to create some eyes).

These two new modifications improved the skill in the game of the players having better results against a human player, indicating that the implementation of more basic rules can improve the player.

CONCLUSION

It was observed that SANE learn more fast how to beat his opponents, but stop the learning when reach the best strategy at some point in the time, by contrary SANEi continues the learning of new strategies to play Go through the generations, which can indicate that SANEi could be a best method to learn how to play Go and not only to beat the opponent. The reason is because SANEi introduce new members into the population in every generation, which give more diversity and the possibility to find new strategies in the game.

In the co-evolution of SANEi against SANE is observed that SANEi beat more frequently to SANE, which indicate that SANEi can produce better strategies in the GO game in the time. The players (networks of neurons) generated using SANEi was tested against human non professional players with better results. These results can indicate as well that the inclusion of an immigration population is other NE techniques can produce good results.

Finally, although SANEi demonstrated better results that SANE, the ultimate goal is to beat to professional Go player, and the future work should be work in new structures and strategies of co-evolution.

BIBLIOGRAPHY

[1] http://www.eurogofed.org/

[2] http://www.nutn.edu.tw/wcci2010/result.htm, Human vs. Computer go Competition, Workshop on Emergent Application of Computational Intelligence in Computer Go.

[3] http://senseis.xmp.net/?MoGo

[4] Computer Go: an AI Oriented Survey, Bruno Bouzy, Tristan Cazenave, University of Paris, 2001.

[5] Applying ESP and Region Specialists to NeuroEvolution for Go, Andres Santiago Perez-Bergquist, 2001

[6] Evolving Neural Network through Aumenting Topologies, Keneeth O. Stanley, Risto Miikkulainen, 2002.

[7] Symbiotic Evolution of Neural Networks in Sequencial Decision Tasks. David Eric Moriarty. PhD dissertation, Department of Computer Sciences of University of Texas,1997.

[8] Learn to play GO. A Master's Guide to the Ultimate Game. Janice Kim (3 dan), Jeong Soo-hyun (9 dan).

[9] Solving Non-Markovian Control Task with Neuroevolution. Faustino J. Gomez, Risto Miikkulainen, Universidad of Texas, 1999.

[10] A comparison between Cellular Encoding and Direct Encoding for Genetic Neuroal Networks, Frederic Gruau, Darrey Whitley, Larry Pyeatt, 1996.

[11] Co-evolving a Go-player neuronal network. Alex Lubberts, Risto Miikkulainen, 2001.

[12] Forming Neural Networks through Eficient and Adaptive Coevolution,David E. Moriarty, Risto Miikkulainen, Evolutionary Computation 1998.

[13] Efficient Reinforcement Learning through Symbiotic Evolution, David E. Moriarty, Risto Miikkulainen, Machine Learning, 1996.

[14] New Methods for Competitive Coevolution, Christopher D. Rosin, Richard K. Belew, 1996.

[15] Arms races between and within species, R Dawkins, J.R. Krebs, Proceedings of the Royal Society of London, 1979.

[16] Monte Carlo Go, Bernd Brügmann, Max-Planck-institute of physics, 1993.