Evolving a Roving Eye for Go

Kenneth O. Stanley and Risto Miikkulainen

Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78705
{kstanley, risto}@cs.utexas.edu
www.cs.utexas.edu/users/{kstanley, risto}

Abstract. Go remains a challenge for artificial intelligence. Currently, most machine learning methods tackle Go by playing on a specific fixed board size, usually smaller than the standard 19×19 board of the complete game. Because such techniques are designed to process only one board size, the knowledge gained through experience cannot be applied on larger boards. In this paper, a *roving eye* neural network is evolved to solve this problem. The network has a small input field that can scan boards of *any* size. Experiments demonstrate that (1) The same roving eye architecture can play on different board sizes, and (2) experience gained by playing on a small board provides an advantage for further learning on a larger board. These results suggest a potentially powerful new methodology for computer Go: It may be possible to scale up by learning on incrementally larger boards, each time building on knowledge acquired on the prior board.

1 Introduction

The performance of artificial intelligence (AI) methods in Go lags behind other board board games, which makes it a popular and challenging testbed for different techniques [1]. Since designing a strategy of a good player by hand is very difficult, machine learning (ML), i.e. letting the computer learn how to play through experience, is a popular approach [2]. The idea is that a sufficiently powerful ML algorithm can learn strategies and tactics through experience that are otherwise difficult to formalize in a set of rules.

The long-term goal is to train an AI player on the full-sized 19×19 board. However, the game is increasingly difficult on larger boards because the search space grows combinatorially and position evaluation becomes more difficult. Thus, current methods [3,4] have been successful only on smaller boards. Such results demonstrate the potential of a method, with the hope that it may be possible to scale it up to larger boards in the future.

Ideally the knowledge gained on small boards could bootstrap play on larger ones. A process that could make use of such shared knowledge would be a significant step towards creating learners that can scale.

Such a learner is presented in this paper, based on the NeuroEvolution of Augmenting Topologies (NEAT) method of neuroevolution [5], i.e. the evolution of artificial neural networks of varying complexity. Previous neuroevolution work in evolving Go focused on networks with the number of inputs and outputs chosen for a particular board size, making it difficult or impossible to transfer to a larger board [3,4]. A different approach is taken in this paper; instead of a network that sees the entire board at once, a *roving eye* is evolved with a small visual field that can scan the board at will. While scanning, the roving eye decides when and where to place a piece. Because it has the same field size on *any* board size, a roving eye evolved on a small board can be further evolved on a larger board without loss of generality. Thus, the roving eye architecture promises to fulfill the goal of a scalable learner.

In order to demonstrate the scalability of the roving eye, a competent Go player was evolved on a 5×5 board against Gnugo, a publicly available Go playing program (available at `www.gnu.org/software/gnugo/gnugo.html`). This neural network was then further evolved against Gnugo on a 7×7 board. For comparison, other 7×7 players were separately evolved from scratch. The main result is that the networks pre-evolved on the 5×5 board improved their performance significantly faster than networks evolved from scratch, and reached a higher final level of performance. This result establishes that (1) the roving eye is a scalable architecture, and (2) scaling can lead to better performance than learning directly on a larger board.

In the next section prior work in machine learning and neuroevolution for Go is reviewed, and also prior implementations of "eyes" in AI. Section 3 then briefly describes the NEAT method for evolving neural network topologies and weights. Finally, Section 5 describes the experimental methods and results.

2 Background

While Go is difficult to master, machine learning techniques show promise. However, current methods cannot scale learning from one board size to another. A roving eye can potentially provide such a scaling capability. Although roving eye systems are not currently used in board games, they are commonly used in robotics. This section reviews the current state of machine learning and Go, and prior applications of roving eyes.

2.1 Machine Learning and Go

Go is a difficult two-player game with simple rules, making it an appealing domain for testing machine learning techniques. The standard game is played on a 19×19 grid. Black and white pieces are placed alternately at intersection points on the grid by the two players. The game ends when both players pass, which usually happens when it becomes clear that no further gains can be made. The winner is determined by the final score.

The object of the game is to control more territory than the opponent. Any area completely surrounded by one player's stones is counted as that player's territory. If the opponent's stones are completely surrounded, those stones are lost and that area is counted towards the other player. If there is an open intersection, called an *eye*, in the middle of the opponent's stones, that intersection must also be filled in order to surround the stones. A structure with two eyes cannot be captured because it is not possible to fill both eyes at once. The *ko rule*, which forbids the same board state from occurring twice, ensures that the game progresses to a conclusion.

The rules of Go are deceptively simple. Yet unlike in many other board games, such as Othello and chess, machines cannot come close to master level performance in Go. Not only are there generally more moves possible in Go than other two-player, complete information, zero-sum, games, but it is also difficult to formulate an accurate evaluation function for board positions [1]. However, the game can be simplified by playing on a smaller board [4]. The smaller the board, the simpler the strategy of the game. At the smallest possible board size, 5×5, the game becomes largely a contest to control one side of the board. However, even at this very small scale, fundamental concepts must be applied, such as forming a line of connected pieces and defending the center. Although these concepts alone are not sufficient to play well on a larger board, they are nevertheless a foundation for more developed strategies, making smaller boards a viable platform for testing machine learning methods. While 5×5 Go may be much simpler than 19×19 Go, it is still related to 7×7 Go, which is related to 9×9 Go, and so on.

Bouzy and Cazenave [1] reviewed a number of general AI techniques not based on learning that have been applied to Go, such as goal generation, game tree search, and pattern-matching [6]. Gnugo 3.2 is a publicly available, open source Go playing program that includes many of these techniques. Gnugo is on par with current commercially available Go playing programs. However, no existing program is even competitive with an amateur shodan, which is the designation for a technically proficient human player [7].

Writing a program to play Go directly is difficult because a large amount of knowledge must be coded into the system. Therefore, machine learning methods that generate their own knowledge through experience are an appealing alternative. For example, Enzenberger [2] created a program called NeuroGo that links units in a neural network corresponding to relations between intersections on a Go board. In 2001, NeuroGo ranked in the top third of computer Go playing programs [1], showing that machine learning is competitive with hand-coded knowledge.

In this paper, neuroevolution (NE) is used to evolve neural networks to play Go. NE has been applied to Go on smaller board sizes in the past [3,4]. However, these experiments evolved neural networks for a single board size, wherein each intersection on the board was represented by a discrete set of inputs and outputs. Such representations cannot easily scale to other board sizes because the number of inputs and outputs in the network are only compatible with the single board size for which they were designed. In contrast, in this paper a roving eye neural network is evolved that uses the same number of inputs and outputs regardless of the board size. The next sections reviews prior research on roving eye systems outside the game-playing domain.

2.2 Roving Eyes

A *roving eye* is a visual field smaller than the total relevant image area; such a field must move around the image in order to process its entire contents. They are often used in robots, where they allow successful navigation even with a limited sensory radius [8]. This type of purposeful control of a moving visual field is also sometimes called *active vision* [9].

Instead of hardcoding the control laws that guide the movement of the roving eye, Pierce and Kuipers [10] showed that they can be learned through experience in the world. This research sets a precedent for the automatic design of roving eyes when the proper control or processing rules are unknown.

Most relevant to game playing are experiments where a roving eye must learn to recognize objects that are too big to process all at once. Fullmer and Miikkulainen [11], and separately Nolfi [12], trained neural networks using neuroevolution to control situated robots that had to move around an object in order to determine its identity. Fullmer and Miikkulainen evolved simple "creatures" that move around a shape on a grid and must learn to move onto only certain shapes. In Nolfi's experiment, a robot moves around an object in order to determine if it is a wall or a type of cylinder. In both cases, both the movement control and the final classification action were evolved simultaneously as neural network outputs.

It is such neuroevolved shape-recognizing roving eyes that provide inspiration for using a roving eye in a board game. However, instead of performing a simple classification, the eye must scan the board and then decide where and when to place a piece. For such a technique to work, the controller for the eye must have memory: It must relate different parts of the board to each other even though they cannot be within its field at the same time. In neural networks, memory can be retained through recurrent connections. Of course, such recurrent structures are likely to be very complex for a game like Go, even at the smallest board size. For this reason, the NEAT method for evolving artificial neural networks, which can evolve increasingly complex network topologies [5,13], was used to develop the controller for the roving eye. NEAT is briefly reviewed in the next section.

3 NEAT: NeuroEvolution of Augmenting Topologies

Developing a Go-playing neural network can be seen as a search problem. It is not known how complex such a network needs to be or even what kind of topology it should have. Searching in too large a space, i.e. a space of highly complex networks, would be intractable while searching in too simple a space would limit solution quality. Therefore, the NeuroEvolution of Augmenting Topologies (NEAT) method [5], which automatically evolves network topology to fit the complexity of the problem, is appropriate for this task. NEAT combines the usual search for the appropriate network weights with *complexification* of the network structure. This approach is highly effective: NEAT outperforms other neuroevolution (NE) methods, e.g. on the benchmark double pole balancing task [5,13]. In addition, because NEAT starts with simple networks and expands the search space only when beneficial, it is able to find significantly more complex controllers than fixed-topology evolution, as demonstrated in a robotic strategy-learning domain [14]. These properties make NEAT an attractive method for evolving neural networks in complex tasks. In this section, the NEAT method is briefly reviewed; see e.g. [5,13,14] for a complete description.

NEAT is based on three key ideas. First, evolving network structure requires a flexible genetic encoding. Each genome in NEAT includes a list of *connection genes*, each of which refers to two *node genes* being connected. Each connection gene specifies the in-node, the out-node, the weight of the connection, whether or not the connection gene is expressed (an enable bit), and an *innovation number*, which allows finding corresponding genes during crossover. Mutation can change both connection weights and network structures. Connection weights mutate as in any NE system, with each connection either perturbed or not. Structural mutations, which allow complexity to increase, either add a new connection or a new node to the network. Through mutation, genomes of varying sizes are created, sometimes with completely different connections specified at the same positions.

Each unique gene in the population is assigned a unique innovation number, and the numbers are inherited during crossover. Innovation numbers allow NEAT to perform crossover without the need for expensive topological analysis. Genomes of different organizations and sizes stay compatible throughout evolution, and the problem of matching different topologies [15] is essentially avoided.

Second, NEAT speciates the population, so that individuals compete primarily within their own niches instead of with the population at large. This way, topological innovations are protected and have time to optimize their structure before they have to compete with other niches in the population. The reproduction mechanism for NEAT is *explicit fitness sharing* [16], where organisms in the same species must share the fitness of their niche, preventing any one species from taking over the population.

Third, unlike other systems that evolve network topologies and weights [17, 18] NEAT begins with a uniform population of simple networks with no hidden nodes. New structure is introduced incrementally as structural mutations occur, and only those structures survive that are found to be useful through fitness evaluations. This way, NEAT searches through a minimal number of weight dimensions and finds the appropriate complexity level for the problem.

4 Experimental Methods

The experiments are designed to answer two questions: (1) Is the roving eye a scalable architecture, i.e. can it play on more than one board size? (2) If so, can learning to play on a small board facilitate further evolution on a larger board? Specifically, does a proficient 5×5 player provide a better starting point for evolving a 7×7 player than evolving from scratch?

4.1 Evolving Against Gnugo

Roving eye neural networks were evolved to play black against Gnugo 3.2 with Japanese scoring. Gnugo is deterministic; it always responds the same way to the same moves. Thus, our solutions were not evolved to be general-purpose players (although they did evolve some general skills), but rather to defeat Gnugo. While it is possible to devise a more general-purpose competition, e.g. by using coevolution, playing Gnugo allows easy interpretation of results and clear illustration of scalability against a known benchmark.

Another advantage of Gnugo is that it estimates the score for every move of the game, as opposed to only the last one. This estimate makes fitness calculation more effective, because the quality of play throughout the game can be taken into account, instead of only at the end. Fitness was calculated from the cumulative score estimate as well as the final score as follows:

$$f = 100 - (\frac{2\sum_{i=1}^{n} e_i}{n} + e_f), \tag{1}$$

where e_i is the score estimate on move i, n is the number of moves before the final move, and e_f is the final score. This fitness equation weighs the average estimated score twice as much as the final score, emphasizing the performance over the course of the entire game over the final position. Such a fitness allows selecting promising networks even early in evolution when the network is likely to lose all its pieces. Because Gnugo returns positive scores for white, they must be subtracted from 100 to correlate higher fitnesses with greater success for black.

Neural networks were evolved to control a roving eye in both 5×5 and 7×7 evolution. In half the 7×7 evolution runs, the champion of the 5×5 run was used to seed to initial population, i.e. the 5×5 champion's connection weights were slightly mutated to form the initial population for 7×7 evolution.

A special modification had to be made in order make Gnugo a useful opponent: Because Gnugo will pass as soon as it determines that it cannot win, Gnugo will pass in 5×5 Go as soon as black plays in the middle. Therefore, to force a continuing game, white was forced to play the intersection directly adjacent and right of center on its first move. That way, Gnugo would play a full game. In order to be consistent, white was forced to make the same initial move on the 7×7 board as well. This modification does not affect the generality of the results since the aim is to encourage the roving eye to improve by playing full games against Gnugo, which was effectively accomplished.

4.2 Roving Eye

The roving eye is a neural network evolved with NEAT. The neural network's sensors are loaded with the visual field of the roving eye, and its outputs determine how the eye should move, or stop moving and decide where to place a piece.

The state of the roving eye consists of its position and heading. It can be positioned at any intersection on the board, and be heading either north, south, east, or west. The ability to see the board from different headings allows the roving eye to process symmetrical board configurations the same way. In other words, unlike full-board neural network players, the roving eye does not need to evolve separate components for symmetric positions, greatly reducing the burden on evolution. Instead, the eye can simply turn around to see the board from an identical perspective.

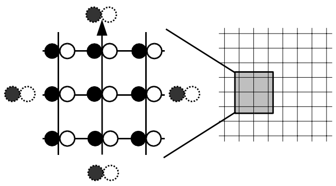

Fig. 1. The Roving Eye Visual Field. At each of the nine intersections visible to the roving eye, it has one black sensor and one white sensor, depicted as filled circles. In addition, the dotted circles represent long-range front, left, right, and back sensors, which only tell how many pieces of each type are in each of those four zones outside the visual field. The arrow shows the eye's current heading. The gray square on the 7×7 board shows the size of the visual field relative to the board. The roving eye also receives 2 inputs representing absolute position, an illegal move sensor, and a bias. This architecture allows the same roving eye to operate on any board size, making scalable Go players possible.

The visual field includes nine intersections with the eye positioned at the center (figure 1). For each intersection, the eye receives two inputs. If the intersection is empty, both are zero. For a black or white stone, the first or second input is fully activated. For a border, both inputs are active. In addition, to help the eye decide where to look, it is given a count of how many white and black pieces are outside its visual field to its front, left, right, and back. The eye is also fed two inputs representing its absolute position on the board, a single input that specifies whether playing in the current position would be illegal due to ko, and a constant bias. Thus, in total, the eye receives 30 inputs regardless of board size.

Five outputs determine the eye's next action. It can place a piece at its current position, move forward in its current heading, turn left, turn right, pause (which may give it more time to "think," i.e. process recurrent activations), or pass. If any of the movement outputs are above 0.5, the eye will move regardless of the other output values, and the movements are combined. For example, the eye can move forward and turn left at the same time. Otherwise, the greatest output over 0.5 is chosen. If no output is sufficiently active, the eye pauses for one time step. If after 100 time steps the eye still does not make a choice, it is forced to pass. Finally, if the roving eye attempts to place a piece on top of another piece, the move is considered a pass.

In general, the roving eye scans the board by moving around until it finds an acceptable location, and then places a piece. In this way, the roving eye analyzes the board similarly to humans, who also focus attention on specific areas of the board while considering a move rather than processing the entire board at once.

4.3 NEAT System Parameters

Because population dynamics can be unpredictable over hundreds of generations, a target of 20 species in the population of 400 networks was assigned to NEAT evolution. The champion of each species with more than five networks was copied into the next generation unchanged. The interspecies mating rate was 0.05. The probability of adding a new node was 0.0045 and the probability of a new link mutation was 0.1. These parameters were found through systematic experimental search. Except when the 5×5 champion was used to start 7×7 evolution, the starting genome had several sensors initially disconnected. That way, NEAT could start in a lower-dimensional space and decide on its own which sensors to use. Specifically, the out-of-range sensors all started out disconnected, and the front-left, front-right, back-left, back-center, and back-right sensors were disconnected from the network. However, these sensors were still present in the genome and were frequently eventually connected to the network by NEAT over the course of evolution.

5 Results

In every run, the roving eye was playing black and Gnugo white. Ten runs of 5×5 evolution were performed. The champion of a typical run, with fitness 99, was chosen as the seed genome for 15 runs of 7×7 evolution. Another 15 runs of 7×7 evolution were started from scratch, without the seed.

5.1 5×5 Champion

NEAT improved significantly against Gnugo in 5×5 evolution. In early generations, NEAT rarely placed a piece without losing it soon after. By the 400th generation, NEAT was able to control the center of the board and thereby capture more area than Gnugo. NEAT learned the general principle of connectedness, and also the importance of maintaining a forward front. The champion roving eye, whose game is shown in figure 2, was used as the seed for further evolution on the 7×7 board.

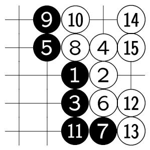

Fig. 2. A Game by the 5×5 Champion. The roving eye (black) is able to control more of the board by pushing its border as far to the right as possible against Gnugo. Gnugo is unable to mount an attack, and instead reinforces its position by creating two eyes. This roving eye was used as the starting point for evolution on the larger 7×7 board.

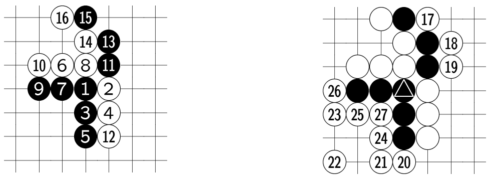

Fig. 3. The 5×5 Champion Plays Gnugo on a 7×7 Board. (a) The roving eye (black) attempts to form a border as it did in 5×5 Go (figure 2). However, because the board is larger, it only partially succeeds. (b) Never having experienced such a conclusion, black does nothing but pass while Gnugo easily eliminates its opponent. The game shows that the 5×5 roving eye is able to apply some of its knowledge to 7×7 Go, although its former strategy is no longer successful.

Figure 3 shows what happens when the 5×5 champion plays directly against Gnugo on a 7×7 board. Interestingly, the champion shows some of its 5×5 capabilities, forming a semi-contiguous line. However, when its line fails to connect, and is not sufficient to cover the larger 7×7 board, the roving eye is quickly surrounded by Gnugo without making any attempt to respond. This behavior makes sense because in the 5×5 game, as soon as the roving eye finished building its border, Gnugo would not attack and the game would end (see figure 2). However, in 7×7 Go, this strategy is no longer sufficient. The question is whether the knowledge contained in the 5×5 champion will benefit further NEAT evolution on the 7×7 board.

5.2 Evolving 7×7 Players

Evolution on the 7×7 board is indeed significantly more effective starting from the pretrained 5×5 champion than starting from scratch (figure 4). Not only does initial fitness rise significantly faster over the first few generations, but it remains higher throughout the run, suggesting that starting raw may never reach the performance level of a roving eye that has already learned about Go on a smaller board. This result establishes both that (1) the roving eye can be used on more than one board size, and (2) evolving on a smaller board captures information that is useful on a larger board.

During 7×7 evolution, NEAT was able to reorganize and expand the original 5×5 strategy in order to form an effective border (figure 5). While evolving from

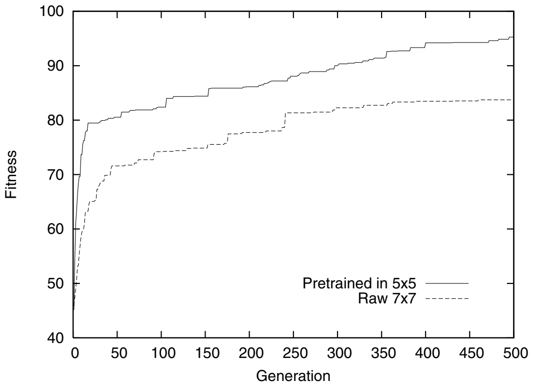

Fig. 4. Average Fitness on a 7×7 Board over Generations. The average fitness (equation 1) of the best roving eye in each generation is shown over 15 runs of raw 7×7 evolution, and 15 runs of 7×7 evolution pretrained in 5×5 Go. Pretrained evolution has significantly ($p < 0.05$) higher fitness in generations 2−237, 383−451, and 495−500, showing that pre-evolving on a smaller board leads to both a higher start and end to evolution.

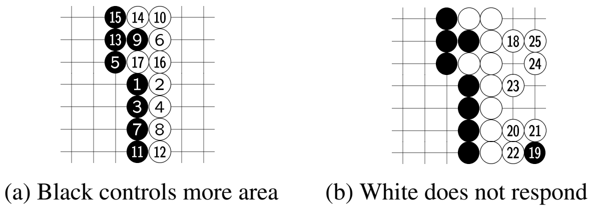

Fig. 5. Typical 7×7 Champion Pretrained in 5×5. The figure shows two halves of a game played by a 7×7 champion with fitness 104 that descended from the 5×5 champion (figure 2). (a) The roving eye has learned to form a clean border with more territory enclosed than Gnugo. (b) The roving eye plays one piece in the corner, because it has evolved to exploit Gnugo's tendency to occasionally place unnecessary pieces (which lowers its score under Japanese rules). Gnugo finishes by creating two eyes, but cannot mount an attack on the roving eye. Thus, the eye finishes with more territory.

scratch required NEAT to discover the capability to connect contiguous intersections, this capability was present from the start when using the pretrained 5×5 champion. In fact, on a larger board, discovering the same early principles takes longer because the game is significantly more complex. Therefore, the raw-starting roving eye was at a disadvantage throughout evolution. In contrast, the pretrained roving eye quickly rises within 25 generations to a level of play that takes raw-starting evolution ten times longer to achieve!

6 Discussion and Future Work

The roving eye allows scaling skills learned on smaller boards to larger boards. This is an important result because current techniques do not succeed in learning to play on larger boards. Therefore, roving eye neuroevolution could turn out to be an important component of a competent learning system for Go.

The level of play demonstrated in this paper is not at championship level. Since black has the first move, it is not surprising that it ultimately controls more area. In addition, it has only learned to play against Gnugo, and would likely fail against a more skilled or significantly different opponent. An important question for the future is how one might evolve a world-class Go player using such a scaling technique. That is, can we apply the roving eye methodology in a way that more general, robust strategies would emerge?

First, the roving eye needs to be evolved on significantly larger boards. There are substantial differences between 7×7 and 19×19 Go: For example, the larger space allows larger patterns to be formed. Evolution will take longer to make use of them, however, rudimentary skills such as the ability to surround enemy pieces or place a contiguous line should still constitute a useful starting point for future learning.

Second, to encourage better generalization, roving eyes could be coevolved instead of evolving them only against Gnugo [7]. Gnugo was used as an opponent in this paper because it is a well-known benchmark, but in the future roving eyes should play against each other to force them to handle a variety of opponent strategies. Well-founded coevolution methodologies such as Host-Parasite, Hall of Fame, or Pareto coevolution should be used to develop the most well-rounded players possible [19,20]. The combination of increasing complexity in NEAT and competitive coevolution has been shown to lead to levels of sophistication unreachable through evolving neural networks of fixed-topology [14], and should work well also in the Go domain.

Third, the roving eye can be combined with game tree search techniques such as α-β. While the state space of Go is too large to be searched directly, a roving eye may help prune the search by acting as a filter that approves or disapproves of searching different positions [21]. In this manner, the search can be made significantly more efficient, and the network does not have to attempt to look ahead to the end of the game. Such hybrid techniques constitute a most promising direction of future research in the long run.

7 Conclusion

The roving eye architecture is an appealing approach to Go because it is the same for any board size. It is also powerful because it can turn to face different directions, allowing it to process symmetrical configurations with the same connections. When compared with evolving from scratch, a 7×7 eye pre-evolved in a 5×5 board achieved significantly faster learning, and significantly higher final fitness. This result establishes that (1) The roving eye can indeed play on different board sizes, and (2) the roving eye aids incremental evolution on increasingly large boards. Thus, the roving eye is a potentially important component of learning systems that aim to perform well on larger boards even when learning directly on such large boards is prohibitively complex.

Acknowledgments. This research was supported in part by the National Science Foundation under grant IIS-0083776 and by the Texas Higher Education Coordinating Board under grant ARP-003658-476-2001. Specials thanks to Darren Izzard for sharing his NEAT-based shape-recognizing roving eye.

References

1. Bouzy, B., Cazenave, T.: Computer Go : An AI oriented survey. Artificial Intelligence Journal **132** (2001) 39–193
2. Enzenberger, M.: Evaluation in go by a neural network using soft segmentation. In: Proceedings of the 10th Advances in Computer Games conference. (2003) 97–108
3. Lubberts, A., Miikkulainen, R.: Co-evolving a go-playing neural network. In: Coevolution: Turning Adaptive Algorithms Upon Themselves, Birds-of-a-Feather Workshop, Genetic and Evolutionary Computation Conf. (GECCO-2001). (2001)
4. Richards, N., Moriarty, D., Miikkulainen, R.: Evolving neural networks to play Go. Applied Intelligence **8** (1998) 85–96
5. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation **10** (2002) 99–127
6. Cazenave, T.: Generation of patterns with external conditions for the game of Go. Advance in Computer Games **9** (2000) 275–293
7. Bayer, A., Bump, D., Denholm, D., Dumonteil, J., Farneback, G., Traber, T., Urvoy, T., Wallin, I.: GNU Go 3.2. Free Software Foundation, Cambridge, MA (2002)
8. Hershberger, D., Burridge, R., Kortenkamp, D., Simmons, R.: Distributed visual servoing with a roving eye. In: Proceedings of the Conference on Intelligent Robots and Systems (IROS). (2000)
9. Dickinson, S.J., Christensen, H.I., Tsotsos, J.K., Olofsson, G.: Active object recognition integrating attention. Computer Vision and Image Understanding **67** (1997) 239–260
10. Pierce, D., Kuipers, B.: Map learning with uninterpreted sensors and effectors. Artificial Intelligence **92** (1997) 169–227
11. Fullmer, B., Miikkulainen, R.: Using marker-based genetic encoding of neural networks to evolve finite-state behaviour. In Varela, F.J., Bourgine, P., eds.: Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life. MIT Press, Cambridge, MA (1992) 255–262
12. Nolfi, S.: Evolving non-trivial behavior on autonomous robots: Adaptation is more powerful than decomposition and integration. In Gomi, T., ed.: Evolutionary Robotics: From Intelligent Robots to Artificial Life. AAI Books, Ontario, Canada (1997)
13. Stanley, K.O., Miikkulainen, R.: Efficient reinforcement learning through evolving neural network topologies. In: Proceedings of the Genetic and Evolutionary Computation Conf. (GECCO-2002), San Francisco, CA, Morgan Kaufmann (2002)

14. Stanley, K.O., Miikkulainen, R.: Competitive coevolution through evolutionary complexification. Journal of Artificial Intelligence Research **21** (2004) In press.
15. Radcliffe, N.J.: Genetic set recombination and its application to neural network topology optimization. Neural computing and applications **1** (1993) 67–90
16. Goldberg, D.E., Richardson, J.: Genetic algorithms with sharing for multimodal function optimization. In Grefenstette, J.J., ed.: Proceedings of the 2nd Intl. Conf. on Genetic Algorithms, San Francisco, CA: Morgan Kaufmann (1987) 148–154
17. Gruau, F., Whitley, D., Pyeatt, L.: A comparison between cellular encoding and direct encoding for genetic neural networks. In Koza, J.R., Goldberg, D.E., Fogel, D.B., Riolo, R.L., eds.: Genetic Programming 1996: Proceedings of the First Annual Conference, Cambridge, MA, MIT Press (1996) 81–89
18. Yao, X.: Evolving artificial neural networks. Proceedings of the IEEE **87** (1999) 1423–1447
19. Rosin, C.D., Belew, R.K.: New methods for competitive evolution. Evolutionary Computation **5** (1997)
20. Ficici, S.G., Pollack, J.B.: Pareto optimality in coevolutionary learning. In Kelemen, J., ed.: Sixth European Conference on Artificial Life, Berlin; New York: Springer-Verlag (2001)
21. Moriarty, D.E., Miikkulainen, R.: Evolving neural networks to focus minimax search. In: Proceedings of the 12th National Conference on Artificial Intelligence, San Francisco, CA: Morgan Kaufmann (1994) 1371–1377