Evolving Artificial Neural Networks through Topological Complexification

Thomas D. Jorgensen, Barry P. Haynes and Charlotte C. F. Norlund

Abstract—**This paper describes a novel methodology for evolving artificial neural network topologies by intelligently adding connections and neurons. The neural networks are complexified and grown to optimise their neural complexity, which is a measure of the information-theoretic complexity of the network. Complexification of neural networks describes the process of increasing the neural complexity whilst increasing the structural complexity of the neural networks. This novel technique is tested in a robot control domain, simulating a racecar. It is shown, that the proposed methodology is a significant improvement over other more common and randomised growing techniques. The technique proposed here helps to discover a network topology that matches the complexity of the problem it is meant to solve. This results in networks which in some cases learn faster than some fixed structure networks.**

Index Terms—**Neural Networks, Complexification, Neural Complexity.**

INTRODUCTION

Artificial Neural Networks (ANNs) have been used in many different applications, with varying success. The success of a neural network, in a given application, depends on a series of different factors, such as ANN topology, learning algorithm and learning epochs. Furthermore all of these factors can be dependent or independent of each other. Network topology is the focus of this research, in that finding the optimum network topology can be a tedious and difficult process. Ideally all network topologies should be able to learn every given task to competency, but in reality a given topology can be a bottleneck and constraint on a system. Selecting the wrong topology can result in a network that cannot learn the task at hand [1]-[3]. It is commonly known that a too small or too large network does not generalise well, i.e. learn a given task to an adequate level. This is due to either too few or too many parameters used to represent a proper and adequate mapping between inputs and outputs.

This paper proposes a methodology that can help find an adequate network topology. The methodology proposes to grow existing networks by adding elements to the network be it connections or neurons, whilst trying to increase a measure of the neural complexity of the network. Assuming that complex task solving requires complex neural controllers, continuously growing and adapting network topology increases the controller complexity and can increase the probability of finding an adequate network topology. The further evolution of an existing network into a more complex one yields an increased chance of better performance and thus a higher fitness.

There are generally 4 ways to construct the topology of an ANN[3]-[5]. (1) Trial and Error, is the simplest method. This essentially consists of choosing a topology at random and testing it, if the network performs in an acceptable way, the network topology is suitable. If the network does not perform satisfactory, select another topology and try it. (2) Expert selection; the network designer decides the topology based on a calculation or experience [3], [6]. (3) Evolving connection weights and topology through complexification. Extra connections and neurons can be added as the evolutionary process proceeds or existing networks can be reorganised [7]-[17]. (4) Simplifying and pruning overly large neural networks, by removing redundant elements [18],[19].

The 4 methods mentioned can be sub-divided into different techniques and approaches, but they all support the fact that continually growing or pruning network topologies yields the most unconstrained and open-ended evolution [15].

This paper starts with a short description of related research in artificial evolution of artificial neural network topology in section 2, followed by a brief description of the neuro-scientific background behind complexification in section 3. Section 4 describes the neural complexity applied in this paper, and section 5 describes in details how it is applied herein. Section 6 describes the results of the experiments conducted here, this is followed by a discussion and a conclusion in sections 7 and 8 respectively.

BACKGROUND

The most common applications of artificial neural networks in both evolutionary robotics and in common AI systems utilize a fixed network structure, in which the connection weights are trained [3]. This fixed structure network is adequate for many different types of systems, and if not, another structure is selected, trained and tested. In systems with inadequate networks, caused by wrong or constraints in network topology, continual complexification of topology by adding valuable components could be the way to find a suitable and adequate topology.

Most research in complexification has so far focused on increasing the structural complexity, i.e. increasing the number of network components, of a neural network, this is done to mimic natural evolution [20]. Different routes and

Manuscript received December 10, 2008.

Thomas D. Jorgensen is with the Department of Electronic and Computer Engineering, University of Portsmouth, Portsmouth, PO1 3DJ, UK (phone +44 2392-842580, e-mail: Thomas.Jorgensen@port.ac.uk).

Barry P. Haynes is with the Department of Electronic and Computer Engineering, University of Portsmouth, Portsmouth, PO1 3DJ, UK (e-mail: Barry.Haynes@port.ac.uk).

Charlotte C. F. Norlund is with the School of Engineering Sciences, University of Southampton, Southampton, SO17 1BJ, UK. (e-mail: ccfn1g08@soton.ac.uk)

techniques have been proposed to continuously complexify neural network for a continuous increase in fitness [8], most prominently is the NEAT framework [15].

Research into the use of neural complexity in complexification to produce biologically plausible structures is limited. This is due to the lack of proper calculation tools and the variety of definitions and focus.

Structural Complexification

The NEAT framework crossbreeds neural networks of different topology. In the NEAT model, mechanisms are introduced to evolve network structure, either by adding neurons or connections, in parallel with the normal evolution of weights. Furthermore different controllers can be crossbred using a gene tracking methodology. The results of these experiments with complexification achieve, in some cases, faster learning as well as a neural network structure capable of solving more complex tasks than produced by normally evolved controllers. One of the main improvements indicated by the success of NEAT is the use of speciation; it increases the search space with only little loss of speed.

Other approaches do not crossbreed networks of different topology, but use mutation as the evolutionary operator that evolves the network. Reference [9], [10] propose networks that are gradually evolved by adding connections or neurons and new components are frozen, so that fitness is not reduced. This is similar to the first method of topological complexification proposed by Fahlman [7], which increased network size by adding neurons.

Neural Complexity

Neural complexity is a measure of how a neural network is both connected and differentiated [16]. It is measure of the structural complexity as well as the differentiated connectivity of the network. The measure was developed to measure the neural complexity of human and animal brains by estimating the integration of functionally segregated modules. This measure reflects the properties that fully connected networks and functionally segregated networks have low complexity, whilst networks that are highly specialised and also well integrated are more functionally complex. Reference [17] has shown that when optimising an artificial neural network with a fixed number of neurons for neural complexity, the fitness increases proportionally, suggesting a link between neural and functional complexity. The more complex a network, the greater the likelihood that it will be capable of solving complex tasks and surviving in complex environments [11]-[17].

NEUROSCIENTIFIC FOUNDATIONS

Complexification in artificial neural networks can prove to be as important, as it is in the development of natural neural systems. It is important in artificial development to unleash fitness potential otherwise left untouched and constrained by a fixed neural topology. Complexification in neural networks is a vital process in the development of the brain in any natural system [21]. Complexification in human brains happens in several different ways, by growth, by pruning and by reorganisation. The first form of complexification happens from before birth and goes on up to adulthood, as the brain is formed. During this period neurons and interconnections grow and hence complexifies the network. The second form of complexification happens through continuous pruning. Connections between neurons have to be used for them not to fade away and eventually possibly disappear. This concept is called neural Darwinism, as it is similar to normal evolution, where the fittest, in this case connections, survive [22]. The third form of complexification happens through reorganisation. In some cases, for yet unknown reasons, connections detach themselves from neuron and reconnects to another. Mostly, reorganisation in natural systems has a detrimental effect.

This paper is only concerned with the first type of complexification. As natural brains are developed during its prenatal phase and during childhood connections and neurons are grown in vast number with a significant amount of redundancy. Research shows that childhood redundancy of neurons and connections and their subsequent pruning is a necessary part of our brains development, as other connections are strengthened after a pruning [21], [22]. Neurons and connections need to be activated and used regularly to prevent them from decaying away, if neurons or connections aren't active they risk decaying away or being pruned. This principle is called “Neural Darwinism”, as only the fittest and most used elements survive, whereas the rest slowly disappear. This methodology tries to avoid growing networks larger than necessary, as this will increase learning time. The aim of this research is to find network topologies that will match the task, it is meant to solve, in size and complexity. Having a too large network will probably mean the task at hand will be solved adequately, but at a cost in form of learning time. Therefore, rather than grow a network with redundancy and then let it be pruned down in size according which elements of the network that are used and which are not, this methodology tries to predict which will be used. This means elements are added if they are likely to be vital elements of the network, i.e. adding the element will probably increase the network's fitness.

THE NEURAL COMPLEXITY MEASURE

The neural complexity measure is an information-theoretic measure of the complexity of the neural network and not a measure of the magnitude of connection weights or of the number of elements in the network [16]. The measure uses the correlation between neurons to quantify the integration and the specialisation of neural groups in a neural network. Neural complexity is a measure of how well brain or artificial neural network neurons have grouped together to form specialised functional clusters and how well these cluster work together. Complex networks have functional clusters that are integrated with each other to work as a unit. This is why complex networks are more likely to be able to solve a complex task. Having a complex network is not a requirement for complex behaviour nor is it a guarantee for complex behaviour, but it increases the chances of finding a neural network that adequately solves a given task. The weights of the connections are all set to a constant value when calculating the neural complexity to avoid basing the neural complexity measure on the

magnitude of the connection weights.

X is a neural system with n neurons, represented by a connection matrix stating where connections go to and from. The information entropy H(X) is used to calculate the integration between components [23], and it is calculated by the standard formula $H(X) = \ln((2\cdot\pi\cdot e)^n\cdot|COV(X)|)/2$. COV(X) is a standard covariance matrix based on the connection matrix and $|\cdot|$ denotes the determinant. The integration between neurons in a system X is defined as:

$$I(X) = \sum_{i=1}^{n} H(x_i) - H(X) \qquad (1)$$

The integration I(X) of segregated neural elements equals the difference between the sum of entropies $H(x_i)$ of all of the individual components x_i of the neural network considered alone and the entropy of the network as a whole.

One of the main features of using the entropy its symmetrical properties. Systems that have highly independent functional components or have highly integrated clusters will have a low complexity. Complex systems are characterised by highly specialised clusters, which are integrated with each other. These properties are desirable, as systems that are fully connected are not particularly complex, because every components is correlated with all other components. Systems that have functional clusters that work independently of each other are also not complex, but only some of its subcomponents are. The integration between components can be calculated in different ways, here it is calculated by equation 2.

$$I(X) = \sum_{i=1}^{n} \ln(2\cdot\pi\cdot e\cdot COV(X)_{ii})/2 - \ln((2\cdot\pi\cdot e)^n\cdot|COV(X)|)/2 \qquad (2)$$

The measure of integration uses a covariance matrix based on the connection matrix. The covariance generally expresses the correlation between two variables. The integration is based on the sum of the natural logarithm of diagonal elements of the covariance matrix. The diagonal elements of the covariance matrix are the variance of the connectivity between components and the off-diagonal elements are the covariance's. The covariance matrix gives a measure of how correlated the different components in a network are, as it is closely related to the correlation matrix. The natural logarithm of the determinant of the covariance matrix is subtracted from the sum of the diagonals yielding the integration of components in a given network. The determinant is here used to penalize great variations in the variance of the connectivity.

The average integration between functionally segregated neural groups with k (out of n) elements is expressed with <I(X)>. j is an index indicating that all possible combinations of subsets with k components are used. The average integration for all subsets with k components is used to calculate the neural complexity:

$$C_N(X) = \sum_{k=1}^{n} [(k/n)\cdot I(X) - \langle I(X_j^k)\rangle] \qquad (3)$$

The neural complexity C_N of a neural system X is the sum of differences between the values of the average integration <I(X)> expected from a linear increase for increasing subset size k and the actual discrete values observed. The complexity of a given system is high when the integration is high and the integration of the any sub-system is lower than expected from a linear increase in subset size. This neural complexity measure yields an estimate of the information-theoretic complexity of a neural network by measuring the integration between individual components and all possible combinations of subsets, which is an improvement over similar methods such as those proposed in [18], [19]. In the case where all components are independent of each other, the integration I(X) = 0 and hence the complexity $C_N(X) = 0$. The integration and the complexity will always be ≥ 0, due to the definitions and nature of the equations. It is worth notion that $\langle I(X_j^k)\rangle$ is monotonically increasing with increasing k. This neural complexity is efficient as it is not based on the magnitude of the connection weights, but on the correlation and variance of the different elements in neural network. The complexity measure has the property that it is not the number of neurons or connections that decides the complexity, but the connectivity. Large networks can be more complex than smaller networks, because it has more elements. Smaller network can be more complex than larger networks as they can be connected better.

USING THE COMPLEXITY MEASURE

The neural complexity measure is used to optimise the complexity of the neural network. A structural addition only takes place if the complexity increases. The complexification methodology proposed is summarised by the following algorithm:

1. Create a starting network, which consists of only the input and output neurons. Furthermore each input neuron should be connected to each output neuron. The starting network used here can be seen Fig. 3.
2. Test the network by evolving connections weights, if the fitness achieved is adequate there is no need to change topology or connection weights, if the fitness isn't acceptable the network should be complexified.
3. Complexify the network by adding elements to the network, this can be neurons or connections.
4. Measure the neural complexity of the network, if the neural complexity has increased the complexification is deemed a success, if it has remained constant or decreased, the changes are undone and the network is re-complexified.

5. The network is trained to competency, if the complexification has increased the fitness, the process has been a success and if desired another complexification can take place. If on other hand the fitness hasn't increased three scenarios exist:
   i. The complexification process is abandoned, as the maximum fitness for the given problem/task has been reached.
   ii. The complexification can be undone, and another different complexification can take. This is done is rare cases, as sometimes certain topologies are unable to adequately learn a given task sufficiently. A different complexification can solve this problem.
   iii. The network can be complexified even further with the existing topology. This is sometimes done as it is assumed that the complexification and the resulting fitness increase has been too small to be statistically significant, a further complexification can increase this significance.

At step 3, when the network is being complexified, there are multiple ways of doing this. Merely adding a number of connections and neurons isn't enough, some of the connections of the original starting network should be replaced by new connections to and from new neurons. This is done to decrease the learning time of the network, as it forces the network topology away from being a reactive network. Adding components can either be a semi-random process or fully optimised one. Randomly replacing some of the existing connections with new ones and randomly adding neurons one by one is recommendable. As long as these additions increase the neural complexity, they are valid. Alternatively, one can try to find the optimum position for new additions, this requires heavy computations as all possible combinations of connections and neurons have to be tested in order to find the optimal configuration.

The Simulated Track and Robot

The controllers evolved here are tested in a simulated environment with a robot. In this environment a robot has to drive around a track, which consists of 32 sections. The objective of this task is to complete 3 laps in the shortest amount of time. If a robot fails to complete 3 laps, the distance covered is the measure of its performance. The robot has to drive around the track covering all of the sections of the track, it is not allowed to skip any sections. In total the robot has to complete 3 laps, with 32 sections in each lap, all visited in the correct order. If the robot is too slow at driving between two sections the simulation is terminated. The following Fig. 1, illustrates the task to be completed:

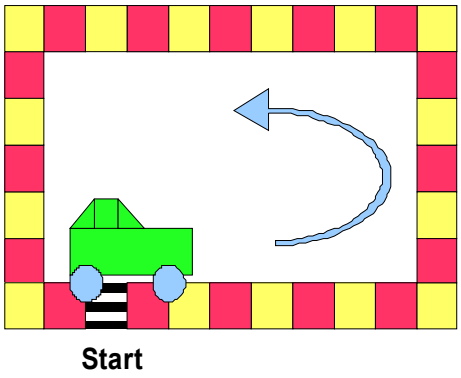

Fig. 1. The figure illustrates the track and the robot in the simulator.

Fig. 1 illustrates the track and the robot driving around it. The robot is not limited in its movement, i.e. it can drive off the track, reverse around the track or adapt to any driving patterns desired, as long at its drives over the right counter clockwise sequence of sections. Fig. 2 illustrates how the robot perceives the track and its environment seen from above.

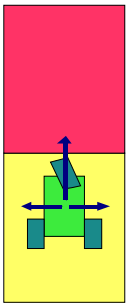

Fig. 2. The figure illustrates the robot and its sensors.

The track sections have alternating colours to mark a clear distinction between sections. The arrows, in Fig. 2, illustrate the three sensors perceived by the robot. The front sensor measures the distance to the next turn and the two side sensor measures the distance to the edge of the track. As indicated by Fig. 2, the simulated robot has three wheels and not four. The reason for this is that it is much more difficult controlling and evolving a controller for a three wheeled robot than for a four wheeled one. The risk when driving this three wheeled robot is, in contrast to a four wheeled vehicle, that it will roll over if driven too abruptly. A robot that has rolled over will not to be able to continue. The front wheel controls the speed as well as the direction. Additionally to the three distance sensors the robot has sensor feedback on its outputs. The robot has a speed sensor and a direction sensor, to measure the actual values of speed and direction, as these may vary from output values from the network. By having these two sensors the network becomes a proper control network with the capability to implement an adequate solution to the task at hand, as well as risking instability.

The Fitness Function

Fitness is rewarded according to normal motorsport rules and practice. 3 laps of the track have to be completed and the controller that finishes in the fastest time wins the race,

i.e. it is the fittest controller. If a controller fails to finish 3 laps, the controller with the most laps or longest distance travelled wins. In the case that two controllers have reached the same distance the racing time determines the fittest controller. The fitness function can in general terms be described by the following:

$$\text{Fitness} = \frac{\text{Distance Covered}}{\text{Time}} \quad (4)$$

The equation states that the longest distance covered in the shortest amount of time yields the best fitness. Time is the time it takes to complete the track. If a controller fails to finish this Time is set to 120 seconds, which is the absolute slowest a controller is allowed to be, before a simulation is stopped. In the likely event that two controllers have covered the same distance, the controller with the fastest time will be favoured for further evolving. The precise version of the fitness function can be seen in the following:

$$\text{Fitness} = \frac{\text{Sections} + (\text{Laps} * \text{Track Length})}{\text{Time}} \quad (5)$$

The fitness is equal to the distance divided by the time. The distance is equal to the number of track sections covered in the current lap, plus the number of sections covered in previous laps. Track length is the total number of sections, which are 32. The minimum fitness obtainable is $1/120 \approx 0.008$, a very god controller can achieve up to 13.

The Test Setup

A total of 4 different tests will be conducted in the experiments. Two of the experiments have fixed network topologies, these are tested to act as comparisons. In one experiment a starting network is evolved randomly by adding neurons and connections to the network whilst evolving the connection weights. Finally, in the last experiement, a starting network is complexified using the neural complexity measure described earlier. The four experiments are briefly summarised in the following:

1. Starting Network, the network shown in Fig. 3 is tested, and it will act a comparison to the here proposed methodology.
2. Benchmark Networks, two standard benchmark networks are tested to contextualise the findings. One network has three hidden layer neurons and the other has five. These networks can be seen in Fig. 4.
3. Randomised Evolution, like the complexification algorithm described previously, the randomised evolution algorithm used here replaces one or more of the connections of the starting network with a random number of new connections and neurons. Up to three successive evolutionary steps take place for the randomly evolution and growth of the starting network.
4. Complexification, using the complexification algorithm described previously, different sets of experiments have been conducted. All experiments differ from one another, due to the randomness of the algorithm. Each experiment conducted with the neural complexity consists of up to three successive changes of the starting network.

The starting network from which all evolved networks have their origin is displayed in Fig. 3.

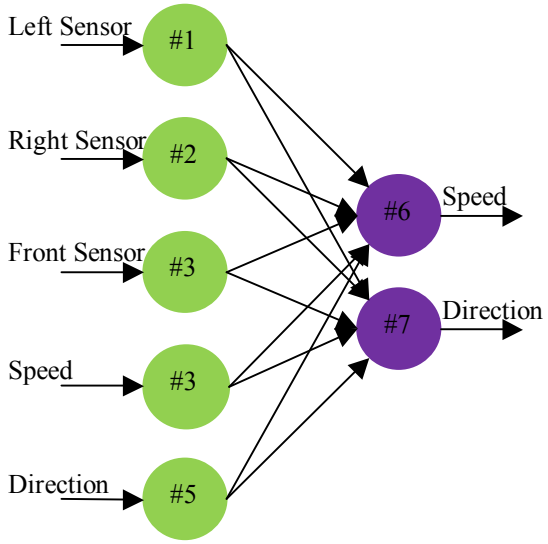

Fig. 3. The starting topology for all evolved network.

The starting network has 5 inputs and 2 outputs, as illustrated in Fig. 3. The inputs are: The three distance sensors, the actual speed and the direction of the control wheel. The outputs from the networks are the speed and the direction of the front wheel of the robot. Having the actual speed and direction of the front wheel as an input means that this system isn't purely reactive, which it is if these are disregarded. Preliminary tests without these two inputs show that a reactive system, consisting of only the starting network, is in most cases, as good as larger network with more connections and neurons.

EXPERIMENTS AND RESULTS

A total of four sets of experiments have been conducted. The first set of experiments was a thorough test of the starting network and its inadequacies. The second set of experiments was with the two benchmark network shown in Fig. 4. In all of these tests the connections weights were evolved and optimised with a genetic algorithm. The third set of experiments was a test of a randomised evolution algorithm. These tests were conducted to prove whether or not the proposed complexification algorithm is efficacious. The final sets of experiments conducted is a thorough test of the here proposed complexification algorithm. The results from the experiments are summarised in Table 1 and described in detail in the following sections.

The Simulation Environment

The evolved neural network controllers are tested in a physics simulator to mimic a real world robot subject to real world forces. The genetic algorithm has in all tests a population size of 25 and the number of tests per method is

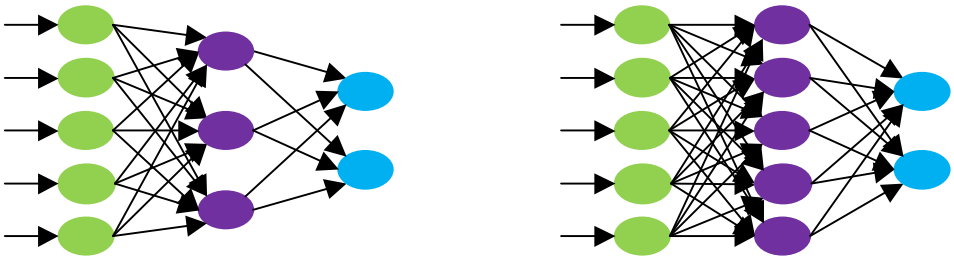

Fig. 4. Benchmark Network 1 (Left) and Benchmark Network 2 (Right).

15. The benchmark networks have been trained for 500 generations, whereas newly evolved networks have been retrained for another 500 generations. The crossover and mutation probabilities were 0.8 and 0.05 respectively. Uniformly distributed noise has been added on the input and output values to simulate sensor drift, actuator response, wheel skid and other real world error parameters. The simulated robot can be seen in Fig. 5, which is a snapshot from the simulator:

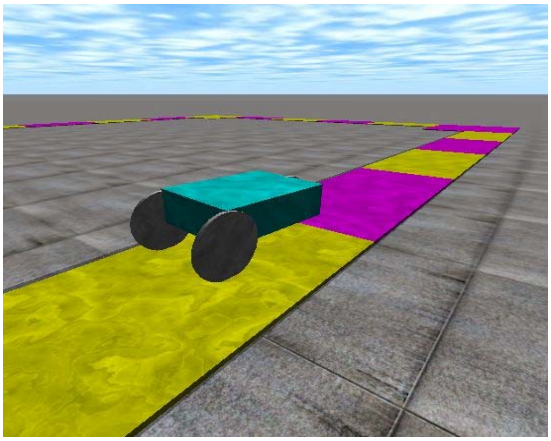

Fig. 5. The robot and the track in the simulator.

The robot, the wheeled box in the middle, is driven along the track, which is visualised by the rectangles of alternating colour. Fig. 5 is similar to Fig. 1 and it gives an impression of how the artificial neural network controls a simulated robot driving around a virtual track. To give the simulation similar attributes and effects as on a real racing track, the track has been given edges, which can be seen in Fig. 5. Whenever the robot drives off the track it falls off this edge onto another slower surface. This means, that if the robot cuts corners, it could potentially have wheels lifting off the ground, thus effecting stability and speed, due to the edge when returning onto the track.

The Starting and Benchmark Networks

The results from the experiments conducted with the starting network clearly illustrate that the network topology is incapable of representing an adequate solution to the problem. One can now either start structurally elaborating this starting topology or select another network topology. The benchmark networks can both adequately represent a controller capable of driving fast around the track. The networks have been selected to make one topology medium sized and another large. This can be seen in Table 1, as the large network has a longer learning time than the medium sized network. Longer learning time can be seen to affect the average fitness of a network, as it simply cannot learn the task given the granted learning time.

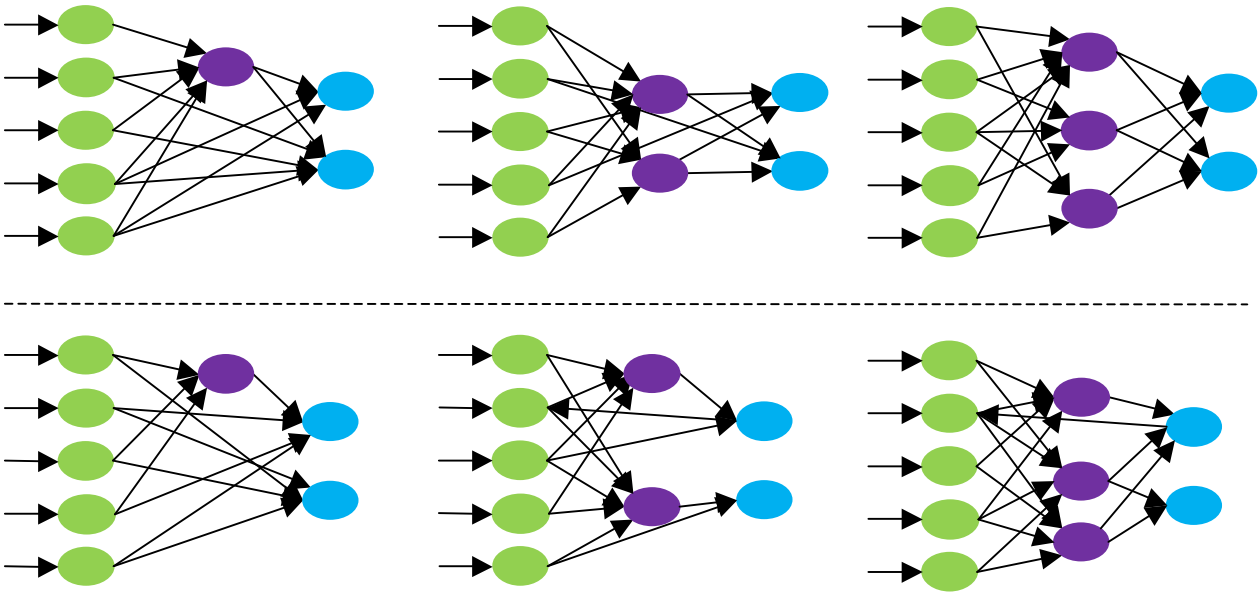

Fig. 6. Randomly evolved networks, steps 1, 2 and 3(Top) and complexified networks, steps 1, 2 and 3(Bottom).

Table 1. Results from the experiments.

Method	Minimum Fitness	Average Fitness	Maximum Fitness	Standard Deviation
Starting Topology	0.01	0.10	0.13	0.06
Benchmark Network 1	7.57	10.45	12.07	1.79
Benchmark Network 2	6.20	9.61	12.86	2.52
Randomised Evolution Step 1	0.08	0.18	0.28	0.09
Randomised Evolution Step 2	4.25	8.26	11.12	2.49
Randomised Evolution Step 3	7.23	10.17	12.29	2.02
Complexification Step 1	0.08	0.18	0.28	0.08
Complexification Step 2	9.02	11.23	12.79	1.41
Complexification Step 3	9.95	10.73	12.56	1.11

Randomised Evolution

The randomly evolved network tests show how the starting network can be evolved into a network that adequately solves the task. Fig. 6. shows an example of how three successive steps of random evolution can evolve the starting network into an adequate one. Each test yields different results and different topologies as a result of the randomness in the algorithm. The different results are averaged and normalised. After one step, some of the initial connections have been replaced by a neuron and some new connections. Despite several new components the network topology is still inadequate and further additions are necessary. After two steps the evolved network can adequately implement the task at hand. The evolved network topologies range from being very capable to inadequate. Some topologies can learn the task fast whereas others cannot learn it at all. After three steps enough components have been added to assure that the task is learned adequately for all evolved networks.

Complexified Networks

The complexification algorithm has been tested thoroughly and the results can be seen in Table 1. The results show some of the same tendencies as the randomised algorithm, no two additions are the same as the number of connections to remove and add is semi-random. The neural complexity increases with each addition to the network, as this is a requirement. The complexity of the starting network is measured to 6.89, this on average increases by 15.32% after the first evolutionary step, 38.12% after the second step and 71.16% after the third. The results show that after the first additions, the network is still incapable of adequately solving the task. After the second step the average network is capable of learning the task at hand fast and competently. The fitness decreases insignificantly after the third step, this is due to the increased learning time after the addition of more components to the network. Had the complexifications stopped after the second step, the algorithm proposed here would have found a network whose topology adequately matches the complexity of the problem to solve. Any further additions beyond this step will not be fruitful.

DISCUSSION AND EVALUATION

The results from all of the experiments show that not all networks are capable of adequately solving the problem. The results furthermore confirm that increasing the neural complexity of a network is no guarantee for success. Sometimes using the complexity measure yields network incapable of solving any given problem. However on average this complexification strategy is more likely to yield good networks than a random evolution strategy. Comparing this methodology with randomised evolution strategy yields interesting results. After one step there is no significant difference between the two algorithms, after two steps the complexification algorithm is on average 26% better than random algorithm, which is a significant difference given a t-test with a 5% significance level. This difference is again insignificant after three steps. This means the complexification algorithm is more likely to find adequate networks to solve the given task. This can be seen by the fact that all complexified networks solve the task and gains a good fitness when solving it. Continually increasing the neural complexity does, for obvious reasons, not always increase the fitness of a network. The difference between the results from the complexifications in the step two and step three is insignificant. In step three the average fitness is lower as a result of the extra components to train. Training very complex networks even further than the networks at step 2 yields no significant improvement in the results. The fitness increase converges after the second step, hereafter any further complexity increases, does not add significantly to the fitness of the network.

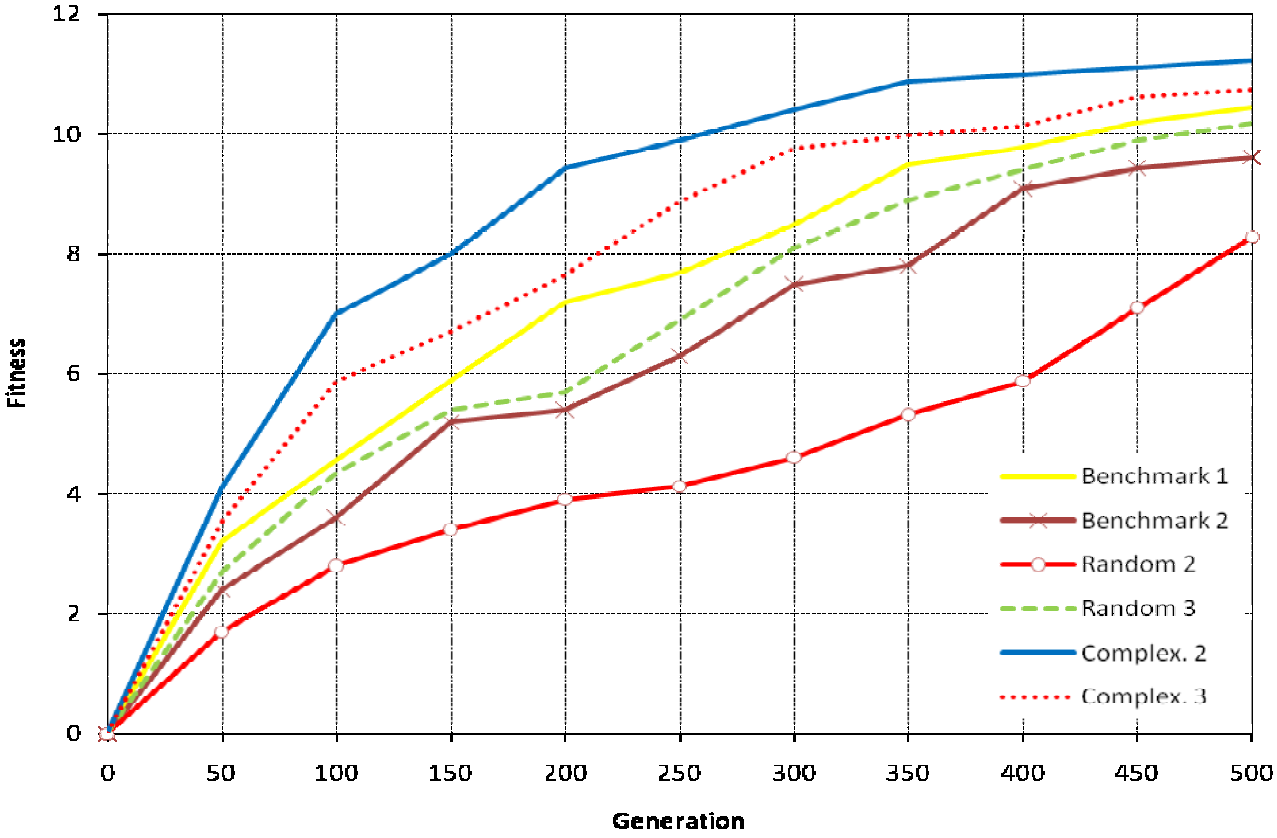

Fig. 7. The average learning speed of the different networks.

The learning speed of the starting network, the benchmark networks, and the newly elaborated networks can be seen in Fig. 7. It is noticeable that all networks loose their fitness immediately after a structural elaboration.

Fig. 8 and Fig. 9 show the route that two different controllers choose to drive around the track. One controller has achieved a high fitness and the other a less than average fitness. The race car starts in (0,0) and drives to (20,0) where it turns. Hereafter it continues to (20,11) where it turns and continues to (-2.5,11) and from here it continues to (-2.5, 0) and on to (0,0). The controller tries to align the car on the straight line between the points. Fig. 8 shows an average lap of a good evolved network, and it clearly illustrates the route that the car takes.

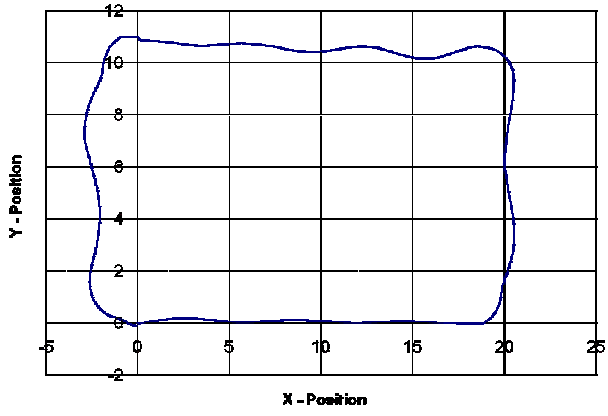

Fig. 8. The route taken by a good controller.

Fig. 8 illustrates how the evolved network performs and the degree of overshoot when turning and recovering to drive straight ahead on another leg of the track. Fig. 9 shows the average route for a poor network for comparison.

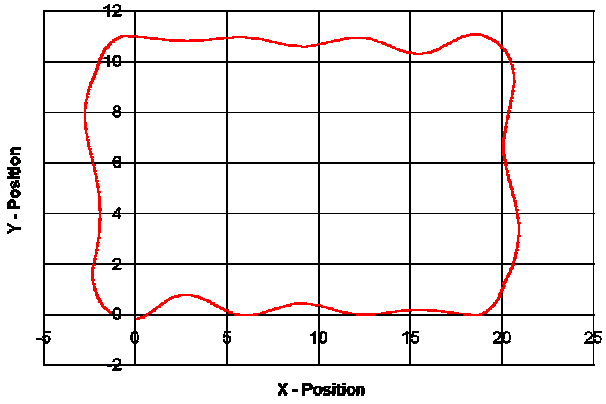

Fig. 9. The route taken by a poor controller.

The two figures show the routes of the different networks. Fig. 9 clearly shows that the poor controller overshoots more than the good network controller in Fig 8. Less overshot, ultimately means that the racing car is able to move faster, which means it has a better fitness. The difference in fitness between the two routes is a factor 2 and the overshoot of the poor controller is more than twice that of the good controller.

Future Work and Direction

The results obtained from the experiments indicate that this methodology is very useful when evolving network

topology and connection weights. Further experiments are to be conducted in order to determine whether this methodology is better than the best growing strategies. This involves implementing successful strategies like NEAT [15]. Further work on this methodology includes making comparisons with the optimum strategy described earlier. This requires different task of different complexity and different starting networks. Different tests and problems to solve will also reconfirm the efficacy of this algorithm.

CONCLUSION

This paper has presented a new methodology for complexifying artificial neural networks through structural addition of neurally complex components. A semi-random number of connections and neurons were added at each complexification step, this was done whilst increasing the neural complexity of the network. A starting network, two benchmark networks, a randomised evolutionary algorithm and the here proposed complexification strategy were tested thoroughly. The results confirm that the here proposed methodology is better than more randomised methods. Complexification and evolutionary strategies that do not take neural complexity into account can prove to be less efficient than this methodology. Using the neural complexity measure and the algorithm proposed herein yields an increased probability of finding neural network controller that adequately solve the here given problem. Using this methodology does not give a guarantee for success, but is does increase the probability of achieving it.

REFERENCES

[1] A.S. Weigend, D.E. Rumelhart and B.A. Huberman, "Back-propagation, weight-elimination and time series prediction," *Proceedings of the 1990 Summer School on Connectionist Models*, 1990.

[2] J. Denker, D. Schwartz, B. Wittner, S. Solla R. Howard, L. Jackel and J. Hopfield, "Large Automatic Learning, Rule Extraction and Generalization," *Complex Systems*, vol. 1, no. 5, 1987.

[3] S. Nolfi, and D. Floreano, *Evolutionary Robotics; The Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT Press 2000.

[4] X. Yao and Y. Liu, "A New Evolutionary System for Evolving Artificial Neural Networks," *IEEE Trans. Neural Networks*, vol. 8, no. 3, May 1997.

[5] X. Yao, "Evolving Artificial Neural Networks," *Proc. of the IEEE*, vol. 87, no. 9, September 1999.

[6] R. Jacobs and M. Jordan, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, 1991.

[7] S. E. Fahlman, and C. Lebiere, "The Cascade-Correlation Learning Architecture," *Advances in Neural Information Processing Systems 2*, Los Altos CA, US, 1990.

[8] T. Jorgensen, and B. Haynes, "Evolving Co-operative behavior," in *Proc. of SAB06 Workshop on Bio-inspired Cooperation and Adaptive Behaviours in Robots*, Rome 2006.

[9] P. Angeline, and J.Pollack, "Evolutionary Module Acquisition," in *Proc. of the Second Annual Conf. on Evolutionary Programmin*g, 1993.

[10] P. Angeline, G. M. Saunders, and J. B. Pollack, "An Evolutionary Algorithm that Constructs Recurrent Neural Network," *IEEE Trans. on Neural Networks*, 1994.

[11] M. Lungarella, and O. Sporns, "Information Self-Structuring: Key Principle for learning and Development," in *Proc. of the fourth IEEE Int. Conf. on Development and Learning*, 2005.

[12] O. Sporns, G. Tononi, and G. M. Edelman, "Connectivity and complexity: the relationship between neuroanatomy and brain dynamics," in *Neural Networks 13*, 2000.

[13] O. Sporns, "Small-world connectivity, motif composition, and complexity of fractal neuronal connections," in Biosystems 85, 2006.

[14] O. Sporns, and M. Lungarella, "Evolving Coordinated Behaviours by Maximizing Informational Structure," in *Proc. of the Tenth Int. Conf. on Artificial Life*, 2006.

[15] K. O. Stanley, and R. Miikkulainen, "Continual Coevolution through Complexification," in *Proc. of the Genetic and Evolutionary Conference*, 2002.

[16] G. Tononi, O. Sporns, and G. M. Edelman, "A Measure for Brain Complexity: Relating Functional Segregation and Integration in the Nervous System," in *Proc. of the National Academy of Science of USA*, May 1994.

[17] L. S. Yaeger, and O. Sporns, "Evolution of Neural Structure and Complexity in a Computational Ecology," in Proc. of the Tenth Int. Conf. on Artificial Life, 2006.

[18] Y. L. Cun, J. S. Denker, and S. A. Solla, "Optimal Brain Damage," *Adv. Neural Inform. Process. Syst. 2*, 1990.

[19] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal Brain Surgeon," *Adv. Neural Inform. Process. Syst. 4*, 1993.

[20] R. Dawkins, *Climbing Mount Improbable*. Reprint by Penguin Books, London, England 2006.

[21] G. N. Martin, *Human Neuropsychology.* Prentice Hall, 1998, Reprinted 1999.

[22] G. Edelman, *Neural Darwinism – The Theory of Neuronal Group Selection*. New York: Basic Books, 1989, Print by Oxford Press 1990.

[23] C. E. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical J.*, vol 27, 1948.