Encouraging Networks Modularity by Seeding Motifs

Shuguang Li[1], Jianping Yuan[1] and Juan Cristóbal Zagal[2]

[1] School of Astronautics, Northwestern Polytechnical University, Xi'an, Shaanxi, P.R. China
[2]Department of Mechanical Engineering, University of Chile, Santiago, Chile
lisg81@gmail.com

Abstract

We propose a motifs seeding method to encourage the emergence of modular structure during network evolution. Previous studies fail to trigger modularity on freeform evolving ANNs either when varying environmental factors or the evolutionary process itself. We extracted statistical profiles of 3-node and 4-node motifs from evolved networks, and then generated new networks by seeding the most useful 3-node motif (feed-forward loop, ID:38). A series of retina recognition experiments was conducted using the seeded networks. The performance of different algorithms was measured. Our results indicate that modularity could be encouraged under certain conditions. We were able to build networks meeting a desired Z-score.

Introduction

Modularity is a common property of natural and artificial complex systems. Networked modular structures commonly arise in biology, computer science, social sciences as well as many other disciplines. One can recognize modularity by the presence of clusters of highly interconnected nodes that are sparsely connected to the remaining ensembles of a networked structure (Newman, 2006). Although it is known that modularity is beneficial for the evolvability and robustness of complex systems, its origin remains to be uncovered. The questions of how modularity emerges in complex systems and how it affects the system's performance during development have been frequently addressed (Wagner and Altenberg, 1996).

Artificial evolution provides an excellent platform for exploring the above questions. A variety of systems have been evolved, including simple equational models, expressed by linear matrix transformations, artificial neural networks (ANNs) (Haykin, 1994), representing complex nonlinear phenomena (Yao, 1999), physical simulations involving complex machines and even real robotic systems (Lipson, 2000).

It is generally believed that modularity should be an outcome of an evolutionary process itself. Some experiments have shown that modularity might speed up an evolutionary process (Lipson et al., 2002). Apparently the mechanisms of selection (adequate choice of fitness function), environment variation and noise generation might play a key role in the emergence of modular structures.

Most of previous models used in artificial evolution are relatively simple. Linear models have been used to simplify the simulations; the nonlinear ANNs models have also been constrained by predefined structures or given building rules. Such limitations could significantly decrease the space of evolutionary search, and simulation results consequently lack of generality.

Freeform ANNs have been employed to increase the generality of systems, but modular networks have not been found under similar experimental conditions at all. Fortunately, further experiments show that the modularity has no conflicts with the evolution of these complex networks (Li and Yuan, 2011). Therefore more effective and general methods have to be designed to encourage the emergence of modularity.

Network motifs are small-scale sub-networks which frequently appear in complex networks, and they have been found in many systems. A network motif can be understood as a pattern or unit of a particular information-processing task. It has been suggested that in many systems the motifs and modularity emerge spontaneously and simultaneously during evolution (Kashtan and Alon, 2005).

Based on these results, we are interested in making use of the coupled mechanism between motifs and modularity, more specifically, in this study we attempt to trigger the appearance of modularity by seeding motifs into ANNs. At first, the motifs' characteristics are extracted from well evolved modular ANNs, and then a series of algorithms is proposed to construct networks with those characteristics. In addition, the well studied retina recognition experiment is conducted in order to make comparisons with previous work.

Background and Previous Work

A common approach to investigate modularity and its effects on complex systems is to use a computer based simulation of an adaptive system. A model represents the system-environment interplay and a fitness function governs species survival. This computer based method can be seen as a simulation of natural evolution.

Lipson et al. presented a linear matrix abstraction of an adaptive system (Lipson et al., 2002). The linear system represents the transformation of resources and functional requirements for the survival of certain life-form. By randomly varying the elements of a matrix representing the environment, it was possible to observe an increase in the system's modularity. The relation between varying rate and modularity has also been studied by experiments. The authors

claimed that modularity arises in evolutionary systems in response to variation.

Following previous suggestions, research on environment variation was further pursued in (Kashtan and Alon, 2005). A simple feed-forward ANN was used to perform the retina pattern classification task, which had limited connections and a small range of weights. The general structural constraints for evolving the networks were also given. The results show that modularity and motifs spontaneously evolved in networks when the goals were switched in a modular manner during evolution. Their later work also suggested that varying environment could speed up the evolution under certain conditions (Kashtan et al., 2007).

To validate whether HyperNEAT could evolve modular neural networks, Clune et al. investigated a series of retina recognition experiments (Clune et al., 2010), which were similar to those used in previous studies (Kashtan and Alon, 2005). Their results show that HyperNEAT has the potential to produce modular structures in some simple cases, but unfortunately it was unsuccessful in more complex problems. In order to enable HyperNEAT to foster modular networks Verbancsics and Stanley presented a seeding method toward local connectivity, which successfully encouraged the natural emergence of modular structures accelerating the simulations as well (Verbancsics and Stanley, 2010).

Instead of changing the environment, HØerstad proposed the method of adding noise to the genotype-phenotype (G-P) mapping (HØerstad, 2010). He used the same retina recognition experiments to test the noised based methods. The ANNs and their encoding method were similar to those used by Kashtan and Alon (2005). Based on a large amount of simulation experiments, he gave a statistical result, showing that the novel method could trigger the appearance of modularity and finally speed up evolution, however, the switch-goal method does not show the same abilities, which are totally against the conclusions of previous study (Kashtan and Alon, 2005).

Recently, a freeform ANN model has been proposed to investigate the mechanism of modularity and their responses to the variation of environment and evolutionary process. Varying scenarios have been experimented, the results show that the evolution performance has been improved in most cases, however, the modularity never appeared among those scenarios. Further experiments show that the proposed networks have the potential to produce modular networks but more advanced methods are still needed to encourage the emergence of modularity on complex networks (Li and Yuan, 2011).

Models, Algorithms and Tools

The ANN Model and Evolutionary Algorithms

To better understand the geometrical properties of complex networks, such as modularity and motifs, we have presented a pure topological ANN (Li et al., 2010), which has binary connection weights and free form directed connections at the hidden layer (Fig. 1). As the architecture shows (Fig. 1), there are three groups of neurons: input neurons, hidden neurons and output neurons, represented as *I*, *H* and *O* respectively.

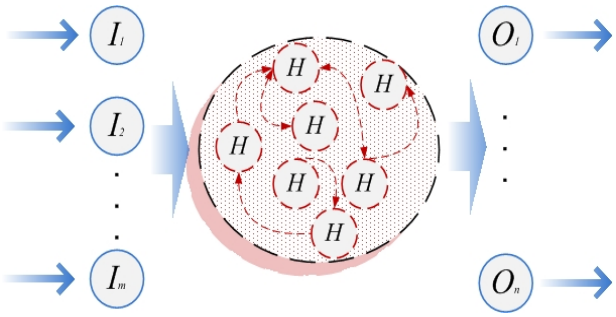


Figure 1: Pure topological neural networks

Therefore, all hidden neurons' and output neurons' values are updated by equation (1) and (2) respectively. The overall model can be given as:

$$H_i(t) = \sin(\sum_{j>i} H_j(t-1) + \sum_{f<i} H_f(t) + \sum_r I_r(t)) \quad (1)$$

$$O_k(t) = (1 + \exp(-\sum_i H_i(t)))^{-1} \quad (2)$$

where H_i denotes the current state of the i^{th} hidden neuron, which is relative to the other hidden neurons (H_j and H_f) and the input neurons (I_r) that connect to H_i. O_k is the k^{th} neuron's state of all n output neurons. Due to the characteristics of activation functions, we need to normalize all input raw data into range of [0, 2π] before computation. Accordingly, we have to scale the output value from [0, 1] to the target range as a final step.

We use the graph encoding method, which directly encodes connections between two nodes in a "from-to" fashion, and then organize all those connections as a graph vector structure. Five evolutionary operators, elitist replication, roulette wheel selection, sub-graph crossover connection mutation and transposing mutation have been have been used to evolve our networks.

Modularity Measurement by Artificial Tracer

We measured the modularity of our ANNs using the artificial tracer method (Li and Yuan, 2011), which is inspired by the chemical, isotopic and radioactive tracers. We created the

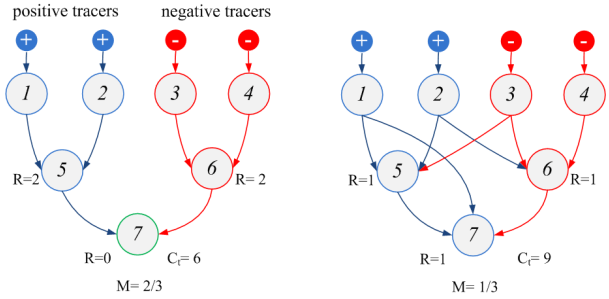


Figure 2: Artificial tracer method

digital tracer elements using different markers, such as positive and negative tracers. To measure the modularity, we first injected the different tracers into each input node of network according to their attributes. All tracers are then passed through other nodes along the directions of information flow. The output connection will pass a tracer to next node with the same marker as the parent node. Annihilation takes place if two tracers with different markers meet at one node. Then we could roughly calculate the modularity using the following equation:

$$M = \frac{\sum_{i=0}^{n} R_i}{C_t} \quad (3)$$

where M represents the degree of network modularity, ranging from 0 to 1. Larger values are assigned to networks with higher degree of modularity. R_i denotes the number of remaining tracers at the i^{th} node after annihilations. We summarize their values as the equivalent of total amount of remaining connections. One should notice that the R_i does not include all input nodes, since the index i starts counting from the first hidden node. C_t is the total number of connections within this network. This computation shows the essence of modularity, which is defined as a relation between the inter-connections and intra-connections of elemental modules. An illustration of this procedure is shown in Fig. 2. It should be noted that this method has a limitation for measuring the feedback loop structure.

The Retina Pattern Recognition Task

We investigated all the scenarios using a classic retina pattern recognition test. The retina pattern recognition experiment has been frequently used in previous studies as a challenging benchmark. Usually, ANNs have been evolved to recognize and classify an artificial retina. Each retina consists of eight pixels (4-pixel wide by 2-pixel height), equally divided into left and right sides, four pixels per side. The goal is to use an ANN to recognize objects in the left and right sides of this retina (Fig.3). As defined in (Kashtan and Alon, 2005), a left object is defined by three or more black pixels or one or two black pixels in the left column only. A right object is defined in a similar way, with one or two black pixels in the right column only. Those eight pixels each could be abstracted as 1 or 0, then those eights binary values could be treated as a group of input signals for the ANN. Finally, the single output (0 or 1) of the ANN is used to decide whether the retina fits the given Boolean logic questions "L AND R", or "L OR R". The "L AND R" is true only if the object exists at both sides of the retina, whereas if the object appears in left side or right side or even both sides, the "L OR R" function is then true.

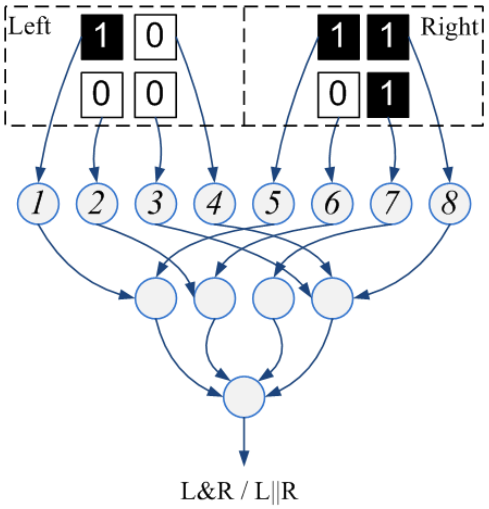


Figure 3: Retina recognition mission

Motifs Analysis and Seeding

We used the software tools Mfinder (Kashtan et al., 2004) and Fanmod (Wernicke and Rasche, 2006) for extracting the motif's feature from evolved networks. MDRAW (Kashtan et al., 2004) was used to display the global network topological architecture. As we know (Kashtan and Alon, 2005), a motif's statistical significance can be described quantitatively using the Z-score.

$$Z_{score} = (N_{real} - N_{rand}) / STD \quad (4)$$

where N_{real} is the number of times the sub-graph appears in the original network, and N_{rand} and STD are the mean and standard deviation of its frequency of appearances in the randomized networks respectively.

Algorithm 1:
SeedMotifs (*Motif_ID, Target_Z-score, Net_size ,Max_refine_times*)

```
1:  Net_pop ← RandomNetworks(Pop_size)
2:  for each Net_i ∈ Net_pop do
3:  App_i ← EnumerateMotifs(Net_i, Motif_ID)
4:  end for
5:  Mean_app ← Average of App_i in Net_pop
6:  STD_app ← Standard deviation of App_i in Net_pop
7:  Target_app ← Target_Z-score*STD_app + Mean_app
8:  Seeding_ model←Initial
9:  Net← Ø; Current_links ← 0; Current_app← 0
10: while Current_links < Net_size do
11:   if Seeding_model= Initial then
12:             Net← MotifSeedInitial (Net, Motif_ID)
13:   else
14:             Net← MotifSeedRefine (Net, Motif_ID)
15:             Refine_count ← Refine_count +1
16:   end if
17:   if Current_links ≥ Net_size then
18:     if Refine_count > Max_refine_times then
19:           return Net
20:     else
21:           Current_app ←EnumerateMotifs (Net, Motif_ID)
22:           if Current_app < Target_app then
23:             Reduce_ratio ← 1- Current_app / Target_app
24:             Net← ReduceLinks (Net, Reduce_ratio)
25:             Seeding_model← Refine
26:           else
27:             return Net
28:           end if
29:     end if
30:   end if
31:   Current_links ← LinksCount(Net).
32: end while
33: return Net
```

Here we propose a series of algorithms to seed motifs into ANNs and then construct the whole network with expected characteristics. We define the network as *Net (N, E)*, where *N* (n_i, $i \in [1, Net\_size]$) is the set of all nodes in this network and *E* (e_t, $t \in [1, Net\_links]$) represents the set of edges. Each edge e_t *{From_node, To_node}* consists of a connection between two nodes. *Net_pop* is defined as a group of *Net*. The *App* indicates the appearance time of specific motifs in network. As the algorithm 1 shows, given the motifs' ID and expected Z-score, the function *SeedMotifs()* is able to construct a network by repeatedly seeding single type motifs. The feedback of current network's motifs could be obtained by calling the function *EnumerateMotifs()*, which will return the appearance time of the motifs, the details of this function are given in algorithm 2. Two types of seeding operators have been designed, which will be used in different stages of seeding. The *MotifSeedInitial()* starts at the beginning of the process, whereas, the refining model *MotifSeedRefine ()* will be executed after reducing the relatively useless links by the *ReduceLinks()*. It should be noted that the Algorithm 2 shown here is just for enumerating 3-node motifs, but it can be easily adapted for detecting other motifs. Other major algorithms could be found at the end of this paper.

Algorithm 2: EnumerateMotifs(*Net*, *Motif_ID*)

```
1:  Net_size ← the size of current Net
2:  En ← the edge number of current Motif
3:  E ← Ø
4:  Motif_app ← 0
5:  for each Ce∈Net.E do
6:      Ce.degree ← 0
7:  end for
8:  for i=1 to En do
9:      Em_i ← false
10: end for
11: for a=1 to Net_size-2 do
12:     for b=a+1 to Net_size-1 do
13:         for c=b+1 to Net_size do
14:             E ← MotifExample(Motif_ID, n_a, n_b, n_c)
15:             for t=1 to En do
16:                 Em_t=EdgeMatch(Net, Me_t)
17:             end for
18:             if all Em_t = true then
19:                 Motif_app ← Motif_app+1
20:                 for each Ce∈Net.E do
21:                     Ce.degree ← Ce.degree +1
22:                 end for
23:             end if
24:         end for
25:     end for
26: end for
27: return Motif_app
```

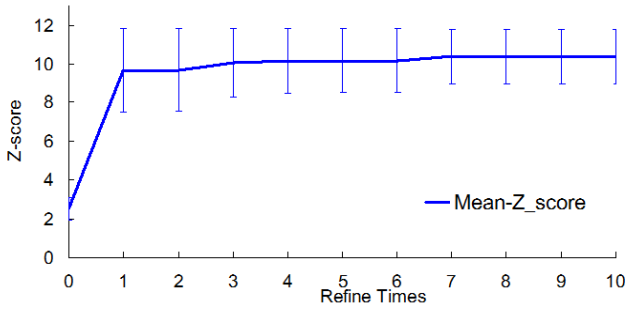
Experiments and Results

Experiments on Performances of Algorithms

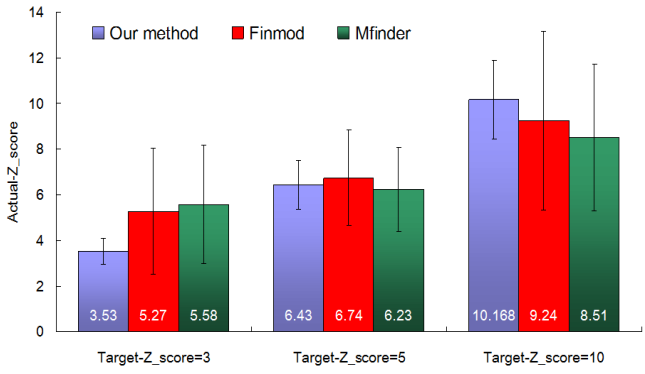
In order to assess the performance of motifs seeding method, we pursued a group of experiments. We used target networks having 30 nodes and 120 links. We focus our study on seeding 3-node motifs, especially the feed-forward loop motif (ID:38). Given a target Z-score of 10 we have executed 10 independent tests to see the capabilities of seeding speed and convergence, the average Z-score and its standard deviation are shown in Fig.4 (a). It is easy to observe that under limited refining times (10), the Z-score mean rapidly approaches to 10 with a small standard deviation. Furthermore, different Z-score requirements have also been tested, and the 10-times average results are compared with other motif detecting tools as shown in Fig.4 (b). As it can be seen the algorithms perform better on the larger Z-score (5 and 10) targets.

Experiments on Motifs Extraction

Before seeding the motifs, we have analyzed the modular ANNs by the Fanmod software. All networks were evolved from our previous experiments (Li and Yuan, 2011), resulting in high values of modularity. We extracted all 3-node motifs and some significant 4-node motifs form 10 networks, the statistical results are shown in Fig. 5 and Table.1. From these results, we could observe some simply statistical attributes among all networks. As for 3-node motifs, the motifs with ID of 38 have a mean Z-score of about 20. This means that motif 38 appears significantly more times than others, whereas the motifs 6, 12 and 36 detected from modular ANNs are less than



(a)



(b)

Figure 4: Performances of algorithms: (a) speed and convergence; (b) accuracy

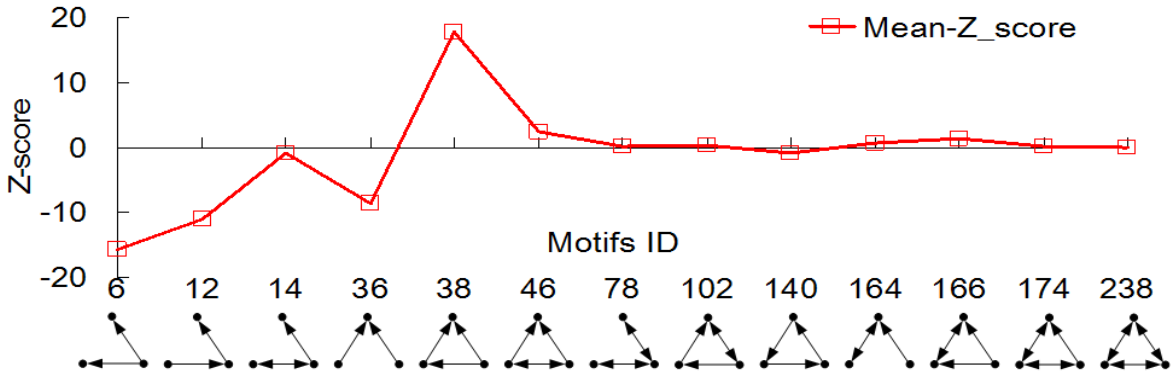


Fig.5. The 3-node motifs' significance profile of networks

	Top two 4-node motifs				Last two 4-node motifs			
Network ID	Motif ID	Z-score	Motif ID	Z-score	Motif ID	Z-score	Motif ID	Z-score
1	2254	9	2252	9.02	2124	-6.73	140	-9.65
2	2254	37.35	2252	20.65	392	-6.71	140	-15.99
3	2254	22.68	2252	18.55	142	-6.79	140	-14.41
4	2254	54.72	2252	40.55	142	-11.89	140	-22.62
5	2254	68.96	2252	46.06	2124	-9.48	140	-24.25
6	2254	6.84	2252	6.18	2124	-4.53	140	-4.88
7	2254	14.26	2252	14.56	2124	-6.92	140	-10.43
8	2254	12.87	2252	11.03	2124	-6.48	140	-12.21
9	2254	7.42	2252	11.13	2124	-5.98	140	-4.362
10	2254	71.7	2252	38.67	2184	-8.49	140	-17.84
Probability	P(2254) =100%		P(2252) =100%		P(2124) =60%		P(140) =100%	
Average Z-score	 ID:2254 A(2254) = 30.58		 ID:2252 A(2252) = 21.64		 ID:2124 A(2124) = -6.67		 ID:140 A(140) = -13.66	

Table 2: The 4-node Motifs' Significance Profile of Networks

in random networks, we name those 3-node motifs as binary tree motif (ID:6), three-chain motif (ID:12) and reverse binary tree motif(ID:36) respectively. Moreover, the 4-node motifs also show very interesting features. The motifs 2254(tetrad-feedforward loops motif) and 2252(bi-feedforward loops motif) appear with highest Z-score among all networks. In contrast, the motif 140(counter-links four-chain motif) seems to emerge much less than others with a smaller mean Z-score of about -13.66

Experiments on Retina Recognition

As for the evolutionary simulation, we first constructed a population of 600 candidate networks by seeding the motifs 38 with a target Z-score of 10 and number of links limited to 120. To reduce the computational complexity, we also constrained the network size to 30 nodes from which 8 nodes were assigned as input pixels' values. One node defined as output, and the remaining nodes (up to 21) were free to build any structures through evolution towards a given task. We set the maximum generation as 5,000 then the modularity was estimated as well as the fitness, and the best networks' structures of each generation were recorded also. In most of our retina recognition experiments, the data set used for training consisted of 100 independent retina patterns which were randomly generated at startup. The general fitness was designed to reflect the ratio of correct recognition over all 100 samples. We evolved the ANNs under a group of different regimes, and we run each test 10 times independently for various experimental scenarios, a list of experiments is shown in Table 1.

We first evolved the networks to recognize the patterns of "L AND R" from the predefined data set. Then, similarly as in previous work, we pursued an interesting MVG regime, in which the recognition goal switched between "L AND R" and

"L OR R" every 50 generations. A varying environment regime (VE) was also tested. We temporally changed the dimension of the data set as a practical method to introduce environment change. Additionally, following the suggestions of (Lipson et al., 2002), we designed the VS scenario as the variations of selection process, the proportion-based roulette selection mechanism sometimes got a failure during evolution, and then the random selection played a key role for producing offspring. The VM scheme temporally applied the mutational operators after the performance evaluation; it thus reversed the traditional sequence between the selection/replication and the mutation every 50 generations.

Experiment	Description
FG-AND	Evolving networks to solve the fixed goal L AND R
MVG	The goal switched between "L AND R" and "L OR R" every 50 generations.
MVE	The date set changed between 100 samples and randomly selected 50 samples every 50 generations.
VS	The selection mechanism alternated between proportion-based roulette selection and random selection.
VM	The order of mutation operation and selection operation reverse d every 50 generations.
FG-M	Same as FG-AND, but the fitness function coupled with the value of modularity

Table 2: The List of Experiments

The comparisons of results on different regimes are shown in Fig.6, results correspond to the average values over 10 runs. Fig.6 (a) shows the best networks' fitness records over all regimes, as we can see, the MVE exhibit a significant higher fitness than others, and it approaches 0.95 within 4,000 generations, whereas the MVG does not show any advantages either in fitness value or evolution speed, its fitness value stays under 0.9.

Fig.6 (b) presents the resulting modularity estimation results for the best evolved networks of all regimes. The figure shows a result that a highly modular structure (>0.8) which never arose among all previous tests. For most of regimes, the modularity values keep under a low level of 0.5.

Although the mean values of VM cases do not show much advance than others, one of VM tests evolved a highly modular structure and with a high correct ratio about 0.9, the correct ratio and modularity are shown in Fig.6 (c).

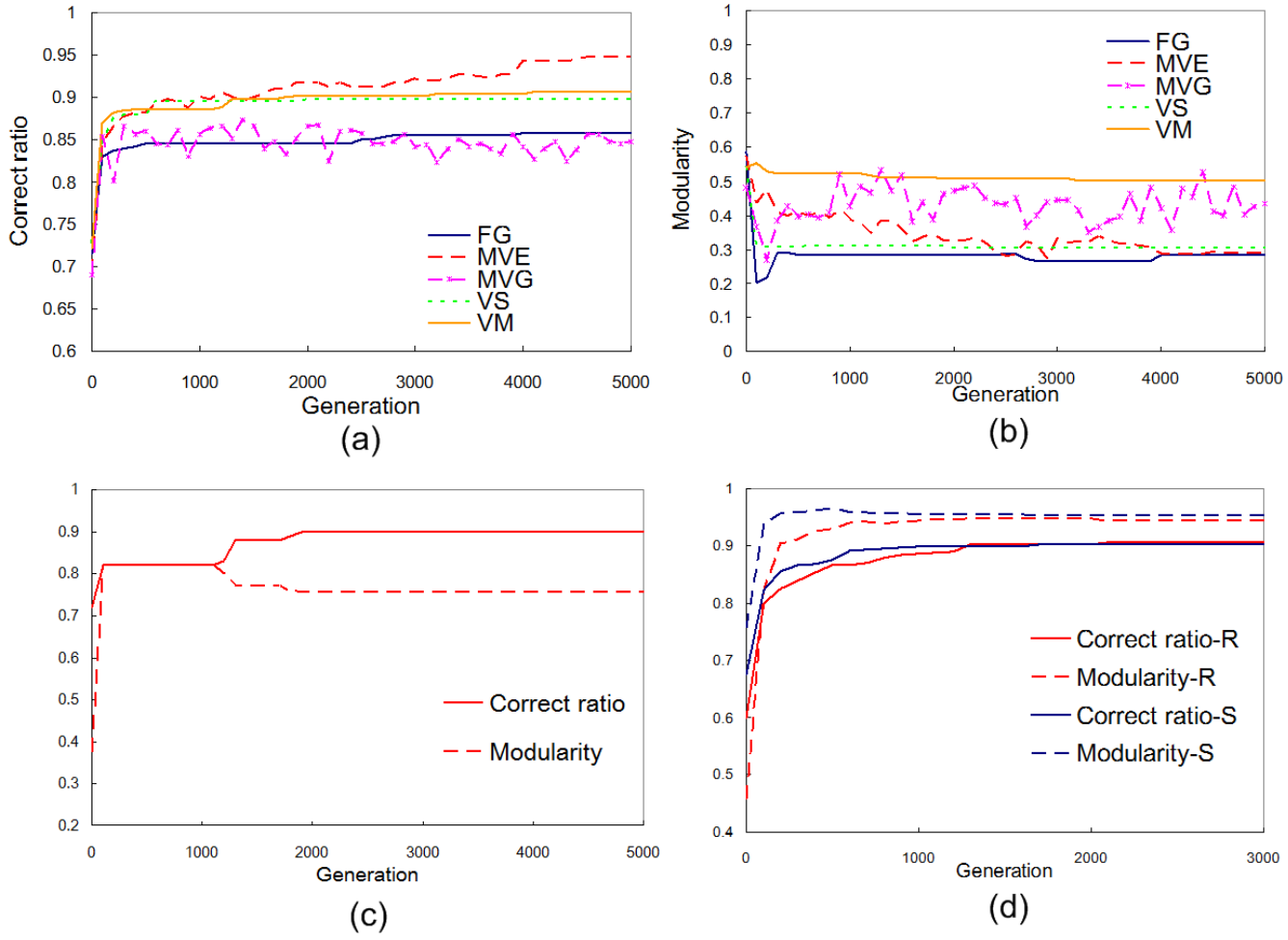


Figure 6: Results of different experiments

As for the FG-M cases, the comparison between random based evolution and motifs-seeding based evolution is shown in Fig.6 (d). It could be easily found that the correction ratio and modularity, they both approach a relative high level at 0.9 and 0.95 respectively. More importantly, these results indicate that the motifs seeding method brings an improvement on the speed of evolution for both correct ratio and modularity.

Discussion

We have proposed a novel method to construct networks by seeding single type motifs. The performances have been tested by two experiments. Compared to other motifs-detecting tools, our method is able to construct networks with predefined Z-score. The seeding algorithms seem relatively accurate for a higher Z-score (≥5) and the target Z-score could be quickly achieved within limited iterations. This is mainly attributed to the operators of refining, after reducing the lowest-degree edges, the remaining edges have the opportunities to be reused in new motifs, and then the density of motifs becomes higher. Since we are aiming at seeding a large population, thus the current method is accurate enough, however, we have to admit that the seeding method still have space to be improved on its accuracy, a real-time feedback mechanism might be useful for a more precise seeding.

After analyzing the well evolved modular networks, we have found that for 3-node motifs, the feed-forward loop motif (ID:38)seems very useful for constructing a modular architecture, but the binary tree motif (ID:6), three-chain motif (ID:12) and reverse binary tree motif(ID:36) conflict with modularity. Similar phenomenon was also found for 4-node motifs, where the tetrad-feedforward loops motif (ID:2254) and bi-feedforward loops motif (ID:2252) always appear much more times than others but the counter-links four-chain motif (ID:140) is useless for a modular structure. These results match with previous work very well, it again validates the idea that motifs could emerge spontaneously as the modularity arises, but the hidden mechanism between them still unrevealed. These phenomenons are probably due to the natural feed-forward information processing of retina recognition tests, and the inherited relations between full-loop structures (motifs) and their sub-structures (motifs).

According to the analysis of our results, we ran various tests after seeding the feed-forward loop motifs into networks, however in most of cases, the modularity of networks have no improvement compared to our previous work. Fortunately, one of the VM tests has evolved a relatively higher modularity than others. As for all the FG-M cases, the performance of modularity and correct ratio have been both improved, the evolved network (Fig.6(d)) presents a nearly perfect modular structure with a high correct ratio. It is obvious that the emergence speed of modular structures is higher than previous results. These results might be attributed to the motifs seeding mechanism, which offers well organized networks for evolution.

Could the motifs seeding method generate highly modular networks regardless the objective of evolution? Since we just simply seed single motif type into a network, the side-effects of seeding have been ignored, however they might be essential for global performances of networks. Based on this hypothesis, the multiple-types or hybrid motifs seeding method is needed in the future study.

Conclusion and Future Work

It is still an open question whether the modularity of ANNs could be encouraged by varying the environment or the evolution process, however, previous work has experimented that the freeform ANNs have difficulty to evolve modular structure under simple variation of external environment.

In this study we try to encourage the networks' modularity by seeding motifs into networks. The motifs statistical features have been extracted from a group of well evolved modular networks. The motif seeding algorithms are proposed and the performances have been evaluated by experiments. We then seeded the network populations by the feed-forward loop motifs and conducted classic retina recognition tests by proposed evolutionary simulation. The modular networks have been discovered during one of tests under varying mutation scenarios. By introducing modularity into fitness function, the modular structures have emerged during evolution; experimental results show that after seeding motifs to initial networks, this emergence process could be accelerated further. These results open the door for triggering modular structure through seeding motifs.

In future, the statistical result will be given based on more experiments under different scenarios. The hybrid motifs seeding algorithms are expected to further encourage the appearance of modularity with a higher success ratio.

Appendices

Algorithm 4: MotifSeedInitial (*Net, Motif_ID*)

```
1:  Net_size ← the size of Net
2:  Success← false
3:  while Success=false do
4:      N ← Random generate different n_a, n_b, n_c
        (a,b,c∈[1, Net_size])
5:      E ← Ø
6:      En ← the edge number of current Motif
7:      for i=1 to En do
8:          Em_i ← false
9:      end for
10:     E ← MotifExample(Motif_ID, n_a, n_b, n_c)
11:     for t=1 to En do
12:         Em_t← EdgeMatch(Net, Me_t)
13:     end for
14:     if all Em_t = true then
15:       Success← true
16:       Net.E ← E
17:     end if
18: end while
19: return Net
```

Algorithm 3: ReduceLinks (*Net, Reduce_ratio*)

1: *Current_links* ← LinksCount(*Net*)
2: *Net.E*← ranked edges by their degrees as a descending order
3: *Reduce_start*←*Current_links** (1-*Reduce_ratio*)
4: **for** *i* = *Reduce_start* to *Current_links* **do**
5: $Ce_i \leftarrow \emptyset$ ($Ce_i \in Net.E$)
6 : **end for**
7: **return** *Net*

Algorithm 5: MotifSeedRefine (*Net, Motif_ID*)

1: *Net_size* ← the size of *Net*
2: *Success*← false
3: *Total_degree*← degree sum of all edges $Ce \in Net.E$
4: **for each** $Ce \in Net.E$ **do**
5: *Ce.s_ratio*← *Ce.degree/ Total_degree*
6: **end for**
7: **while** *Success=false* **do**
8: *Se* ← the edge *Ce* selected by roulette mechanism based on *s_ratio*
9: n_a←*Se.from_node* ; n_b←*Se.to_node;*
10: n_c ← Random generate n_c ($c \in [1, Net\_size]$, $n_c \neq n_a$ or n_b)
11: *E* ← Ø
12: *En* ← the edge number of current *Motif*
13: **for** *i*=1 to *En* **do**
14: Em_i ← false
15: **end for**
16: *E* ← MotifExample(*Motif_ID,* n_a, n_b, n_c)
17: **for** *t*=1 to *En* **do**
18: Em_t=EdgeMatch(*Net,* Me_t)
19: **end for**
20: **if** all Em_t = true **then**
21 *Success*← true
22: *Net.E* ← E
23: **end if**
24: **end while**
25: **return** *Net*

References

Angeline, P. J., Saunders, G. M., and Pollack, J. B. (1993). An evolutionary algorithm that constructs recurrent neural networks. *IEEE Trans. Neural Networks*, 5(1): 54–65.

Clune, J., Beckmann, B. E., McKinley, P. K., and Ofria, C. (2010). Investigating whether hyperneat produces modular neural networks. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 635–642.

Floreano, D., Dürr, P., and Mattiussi, C. (2008). Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1): 47–62.

Haykin., S. (1994). *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ.

Høverstad, B. A. (2011). Noise and the Evolution of Neural Network Modularity. *Artificial Life*, 17(1):18.

Kashtan, N., Itzkovitz, S., Milo, R., and Alon, U. (2004). Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20(11): 1746–1758.

Kashtan, N., and Alon, U. (2005). Spontaneous Evolution of Modularity and Network Motifs. *Proceeding of National Academy of Sciences*, 102(39):13773–13778.

Kashtan, N., Noor, E., and Alon, U. (2007). Varying environments can speed up evolution. *Proceeding of National Academy of Sciences*, 104(34): 13711–13716.

Li, S., Yuan, J., Yue, X., and Luo, J. (2010). The binary-weights neural network for robot control. *Proceedings of the International Conference on Biomedical Robotics and Biomechatronics (BioRob 2010)*, pages 765–770.

Li, S., and Yuan, J. (2011). The Modularity in Freeform Evolving Neural Networks. *Proceedings of the IEEE Congress on Evolutionary Computation 2011*. pages 2593–2598.

Lipson, H., Pollack, J.B. (2000). Automated Design and Manufacture of Artificial Lifeforms. *Nature*, 406: 974–978.

Lipson, H., Pollack, J.B., and Suh, N.P. (2002). On the Origin of Modular Variation. *Evolution*, 56(8):1549–1556.

Moriarty, D. E. and Miikkulainen, R. (1997). Forming Neural Networks Through Efficient and Adaptive Coevolution. *Evolutionary Computation*, 5(4): 373–399.

Newman, M. E. J. (2006). Modularity and community structure in networks. *Proceeding of National Academy of Sciences*, 103(23): 8577–8582.

Verbancsics, P., and Stanley, K. O. (2010). Constraining Connectivity to Encourage Modularity in HyperNEAT, Technical Report, CS-TR-10-10, Department of Electrical Engineering and Computer Science, University of Central Florida.

Wagner, G. P. and Altenberg, L. (1996). Complex adaptations and the evolution of evolvability. *Evolution*, 50: 967–976.

Wernicke, S. and Rasche, F. (2006). FANMOD: a tool for fast network motif detection. *Bioinformatics*, 22(9): 1152–1153.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, 87:1423–1447.