

An Evolvability-enhanced Artificial Embryogeny for Generating Network Structures

Hidenori Komatsu

Central Research Institute of Electric Power
Industry
2-11-1, Iwado Kita, Komae City, Tokyo, JAPAN
komatsu@criepi.denken.or.jp

Yasuhiro Hashimoto, Yu Chen and Hirotsada Ohashi

Dept. of Systems Innovation, School of Engineering,
University of Tokyo
7-3-1, Hongo, Bunkyo-ku, Tokyo, Japan
{hy, chen, ohashi}@sys.t.u-tokyo.ac.jp

ABSTRACT

Existing Artificial Embryogeny (AE) models are insufficient to generate a network structure because the possible links are limited to those connecting nodes with their predefined neighbors. We propose a novel network generating AE model capable of generating links connected to predefined neighbors as well those to non-neighbors. This mechanism provides additional flexibility in phenotypes than existing AE models. Our AE model also incorporates a heterogeneous mutation mechanism to accelerate the convergence to a high fitness value or enhance the evolvability. We conduct experiments to generate a typical 2D grid pattern as well as a robot with a network structure consisting of masses, springs and muscles. In both tasks, results show that our AE model has higher evolvability, sufficient to search a larger space than that of conventional AE models bounded by local neighborhood relationships.

Categories and Subject Descriptors

I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search – *Plan execution, formation, and generation*. F.1.1 [Computation by Abstract Devices]: Models of Computation – *Self Modifying Machine*.

General Terms

Algorithms, Design, Experimentation.

Keywords

Evolutionary Algorithm, Artificial Embryogeny, Mutation, Network Structure

1. INTRODUCTION

Artificial Embryogeny (AE) is a strategy of evolutionary computation inspired by the embryogeny of natural organisms. The primary characteristic of AE is the use of the same genes

multiple times in the process of building a phenotype. Such gene reuse allows compact representations of complex phenotypes [15]. Over the decade, certain AE model variants have been proposed. Eggenberger modeled embryonic dynamics considering the transcription factors [2]. Bentley et al. studied various types of AE implementations for the tessellation problem [1]. Miller et al. proposed multicellular simulated organisms with high robustness at a phenotypic level using the Cartesian Genetic Programming (CGP) method [11, 12].

Although AE is successful in generating structures with certain specific functions such as tessellating tiles or virtual robots, several issues have been pointed out. For example, Harding et al. [4] and Viswanathan et al. [17] reported that poorly encoded AE can retard evolution. Stanley pointed out the overabundant factorization and brittle modularity of the AE model [14]. The issues we would like to highlight relate to the expressive power of the AE model and its evolvability.

Firstly, most of the AE models focused solely on the generation of simple 2D grid patterns. However, the number of real world problems modeled by such AE models is limited, while some AE models already exist that are capable of generating not 2D grid patterns but more general network structures, such as trusses [7, 8] or virtual robots [5]. However, these models are so dependent on predefined local neighbors of nodes. We need a mechanism to extend the predefined search space bounded by the predefined local neighborhood relationship, in order to find a network with higher fitness.

However, this may result in slower convergence. It highlights the issue of evolvability. In this paper we use the term evolvability as an ability to converge fast to a high fitness value, while it has a more general meaning.

Secondly, in the AE research field, few discussions exist concerning the insufficiency of evolvability [9]. In the direct coding research field, a mutation mechanism to allow the mutation rate to be adaptive for each locus is shown to be efficient in terms of evolvability [16]. This kind of approach is also considered efficient for the developmental process of the AE model.

In this paper, we propose a novel AE model, capable of generating network structures. Our AE model involves two novel mechanisms. The first is to generate links between nodes in such a relationship that transcends a predefined neighborhood relationship. The second is a novel mutation mechanism called

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee, provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GECCO '08, July 12–16, 2008, Atlanta, Georgia, USA.

Copyright 2008 ACM 978-1-60558-130-9/08/07...\$5.00.

“heterogeneous mutation” to enhance evolvability. Examples are shown in two different tasks. One is the generation of a 2D grid pattern, and the other is the generation of a robot called “Sodaplay robot”, whose structures can be described as a network [13]. These experiments have two purposes. The former is to show that our model is efficient in solving frequently tackled target pattern generation, while the latter is to show that our model can be applied to the generation of structures to which existing AE models are inapplicable.

In both experiments of 2D grid pattern generation and generation of Sodaplay robot, we also show that our novel mutation mechanism significantly improves the evolvability of the model.

The remainder of this paper is organized as follows. In the following section, we review related works based on AE concerning network structure generation. Section 3 describes our novel AE implementation for the generation of network structures. Section 4 reports on experiments involving the evolution of a 2D grid pattern. Then in Section 5, a “Sodaplay robot” is evolved. Section 6 provides discussion, and Section 7 concludes and presents further work.

2. RELATED WORKS

Recent research in the AE field has begun to focus not only on its fundamental characteristics but also on various kinds of application [3, 5, 6, 7, 8]. In particular, trusses are thoroughly studied as an example task of designing network structures.

Kicinger et al. generated tall and robust buildings with truss structures by evolving cellular automata rules and the initial states as genes [6]. In this approach, trusses are generated as lattice structures with diagonal links. Therein, each cell is transformed into a rectangular-shaped truss filled with predefined combinations of trusses, which is equivalent to solving only a combination problem when the local structures are given.

Kowaliw et al. presented a novel AE model called a Deva 1.N model, which can evolve network structures, and applied it to the generation of truss structures [7, 8]. In the Deva 1.N model, trusses are generated in the five directions of up, left, right, upper-left and upper-right via the bit transformation of 32 types of cell arranged in lattice form. Deva 1.N also involves an operator generating longer links, which deviate from the local neighborhood relationship defined by the lattice structure. However, the Deva 1.N framework is insufficient to search for the optimal network structures because the maximum number of links per node is limited by the definition of a local neighborhood.

Other AE applications are for neural networks or virtual robots. Federici showed that the generation of neural controllers for agents based on an AE model is effective in finding a strategy to collect food efficiently [3]. Hornby et al. showed that virtual robots evolved with an AE approach can attain higher fitness than robots generated with direct coding [5].

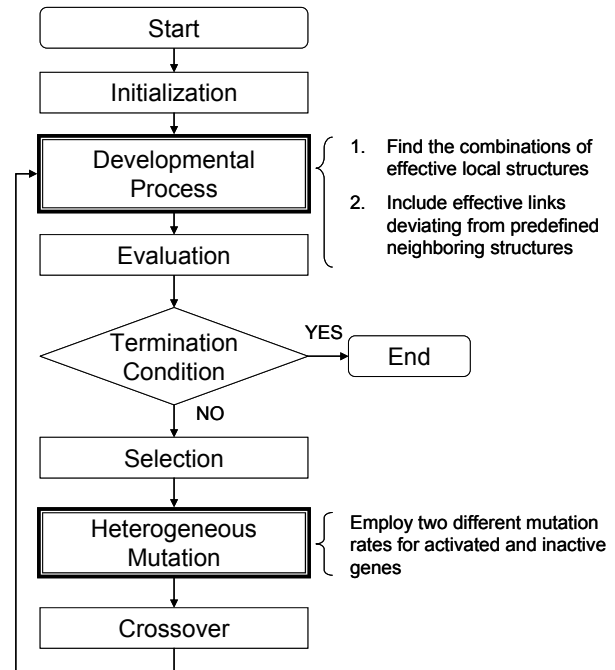


Figure 1: The flowchart of our AE model.

3. NETWORK GENERATING ARTIFICIAL EMBRYOGENY

In this section, we propose an AE model of high evolvability for the generation of network structures.

Implicit Embryogeny (IE) is a kind of AE and was originally proposed for generating simple patterns in the 2D grid [1]. IE is suitable for an understanding of the AE framework, because it is a simple and highly abstract developmental model. We extend this framework of IE in order to build a novel AE model for generating network structures; hence allowing a greater variety of problems to be modeled.

In this paper, we explain our AE model for 2D phenotypic space for reasons of simplicity and demonstrate its ability by generating a Sodaplay robot simulated within a 2D physical space. However, our AE model can be easily extended to generate a network structure in a higher dimensional space.

While many possible variations exist for implementation, the eccenses of our AE model include the following two elements: The first is a mechanism to generate links between nodes in such a relationship that transcends a limited neighborhood relationship. The second is a novel mutation model, namely “heterogeneous mutation”, to enhance the evolvability of the model. The flowchart of our AE model is shown in Figure 1.

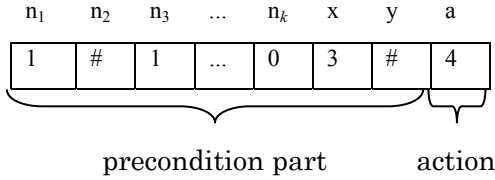


Figure 2: A gene for generating network structures.

3.1 Network Generating Framework

In the network generation by IE-like models, a node is mapped to a cell and the neighbors of a cell are the candidates of linked nodes. In our AE model, the neighbors of a node are the “preferential” candidates of linked nodes and we do not exclude the possibility of nodes being linked to others which is not within the predefined neighbors. To enable such linkage, we introduce an action called “reconnection”, which replaces a pair of connected links with a direct link and connects non-neighboring nodes.

In IE, a gene consists of two precondition parts and an action part (Figure 2). One of the precondition parts is for checking the state of neighboring cells, while the other is for the global positional information of the cell (Table 1). An action part is for acting on neighboring cells. This can be regarded as a situation whereby a cell checks both local and global information, whereupon the action is only applied locally. To convert this framework for generating network structures, we implemented the model as follows:

- (1) Firstly, some nodes are generated randomly or arbitrarily within the 2D space (Figure 3).
- (2) A node around the center of the 2D phenotypic space, which is corresponded to “seed,” is chosen and used as the starting point for network development.
- (3) Checks are made to identify in which area of the 2D phenotypic space the node exists. An area is given by dividing the 2D space equally for both x and y axes. Subsequently, the index number for each area is given by integers for x and y like (0,0) or (1,3).
- (4) When the preconditions are met, an action part, which is for connecting or disconnecting a link, or reconnecting two links, is activated (Figure 4). Note that all the precondition parts need not necessarily be matched for a rule to be fired if the number of matched preconditions exceeds a given threshold.
- (5) Each individual in the population has a genome as a sequence of the genes (rules) defined above. Its corresponding network structure as a phenotype is generated by using the genome for a given number of iterations. Subsequently, the evolutionary operators are applied to evolve the population.

Table 1: Elements of a precondition part.

	Description	Value range
n_1	State of the 1st closest node	0, 1, #
n_2	State of the 2nd closest node	0, 1, #
n_3	State of the 3rd closest node	0, 1, #
...
n_k	State of the k th closest node	0, 1, #
x	Index of area for x	0, 1, 2, ..., #
y	Index of area for y	0, 1, 2, ..., #

k : The number of closest nodes regarded as neighbors to a node.

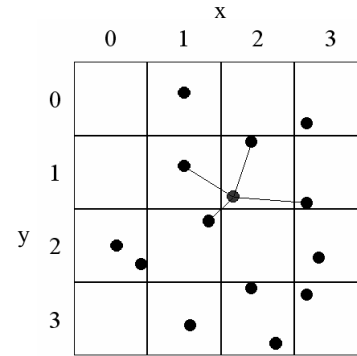


Figure 3: 2D phenotypic space.

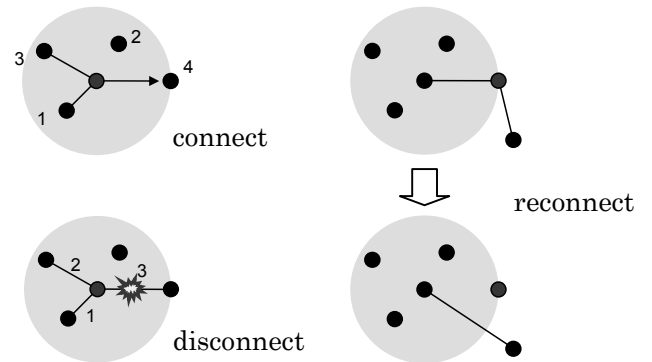


Figure 4: The conceptual diagram of implemented actions.

In the case of pattern generation by conventional IE, a filled cell in a 2D grid represents a living cell and a blank cell represents a dead one. Corresponding to this, nodes with at least one link are regarded as “living” and those without links as “dead”. Moreover, living nodes check the state of neighboring nodes by distinguishing whether they are living or dead. The states of the nodes correspond to the value of n_i in Figure 2 and Table 1 (where 0 is *dead*, 1 is *living*, and # is *don't care*). Note that all these rule activation processes only occur in “living” nodes.

For details concerning connecting or disconnecting a link, see Table 2. If reconnection is applied to a node, the two shortest links are disconnected and then a link is newly generated between the two nodes. Reconnection is activated only when a node has at least two links.

Table 2: Values describing the action.

Value	Action
0	Connect a link to the 1st closest node
...	...
$k-1$	Connect a link to the k th closest node
k	Disconnect the 1st shortest link
...	...
$2*k-1$	Disconnect the k th shortest link
$2*k$	Reconnect the 1st and 2nd shortest links

The reconnecting action is introduced so that links deviating from predefined local neighborhood relationships can be generated. In other words, the target node never has a link to nodes outside the gray circle without a reconnecting action (Figure 4). Note that connection is based on the order of how close the node is while disconnection is based on that of how long the connected link is.

3.2 Heterogeneous Mutation

The evolutionary operators in our AE model are the same as a typical Genetic Algorithm (GA) except for our novel mutation mechanism called “heterogeneous mutation.” When the mutation rate is fixed for every part of a genome, as in the conventional IE, even an individual with maximum fitness includes a considerable number of genes that do not contribute to its developmental process at all.

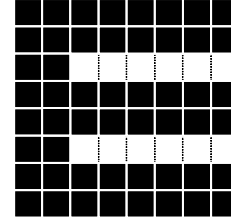
To reduce the number of such unused genes, we introduce a mechanism whereby mutation at a higher rate is applied to genes which are not activated during the developmental process. To implement this heterogeneous mutation, every gene is checked as to whether it was activated or not during the developmental process. Subsequently, mutation is applied to activated genes at a lower rate (usually 0.01) and inactive genes at a higher rate (the appropriate value may differ from task to task, but is 1.0 here). Finally single-point crossover is applied to the population to generate offspring. Note that an inactive gene does not always mutate into an active gene, even if they are mutated at a higher rate. Similarly, an active gene can be mutated into an inactive gene at a lower rate. Our heterogeneous mutation mechanism is not intended to prohibit inactive genes.

4. EVOLUTION OF A 2D GRID PATTERN

In this section, we investigate the effectiveness of our AE model by applying it to a 2D grid pattern generating problem. This problem is frequently tackled by existing AE models.

4.1 Experimental setting

We evolved the population using a network generating AE with heterogeneous mutation, while also using the following three types of AE model for comparison.

**Figure 5: Predefined target pattern “E.”****Table 3: Parameters for the generation of a 2D grid pattern.**

Population size	50
Number of elites	2
Number of genes	20
Number of iterations for growth	8
k (defining neighborhood)	4
Necessary number of preconditions to be matched for a rule to be fired	6
Selection method	Roulette
Crossover method	Single point
Probability of crossover	1.0
Probability of mutation for activated genes	0.01
Probability of mutation for inactive genes	1.0
Division number for the phenotypic space	8*8

- (1) Network generating AE without heterogeneous mutation
- (2) 2D grid AE with heterogeneous mutation
- (3) 2D grid AE (i.e., conventional IE)

The predefined target pattern was set as shown in Figure 5. In the previous section some nodes are generated randomly in 2D phenotypic space, but in this experiment, nodes were positioned at the center of each cell and the seed position was set to the center of the phenotypic space. The parameters were set as shown in Table 3. The fitness for each individual was given by the number of cells in correct states. Thus, the maximum fitness value was 64.

4.2 Results

Figure 6 shows the convergence property of the four types of model by averaging the results of 20 runs. The 2D grid AE with heterogeneous mutation is the best in terms of evolvability, followed by the network generating AE with heterogeneous mutation, with only a slight difference. Figure 7 shows an example of a developmental process with our AE model, which is also capable of successfully generating a 2D grid pattern.

Figure 6 also shows that the evolvability of network generating AEs decreases compared to 2D grid AEs. Considering the cause of this decrease, in a 2D grid AE, the single operation of firing an action part directly corresponds to the development or deletion of a single cell, which, in turn, corresponds to one point of fitness. On the other hand a single operation in a network generating AE has an effect on two nodes, because a link is connected to two nodes.

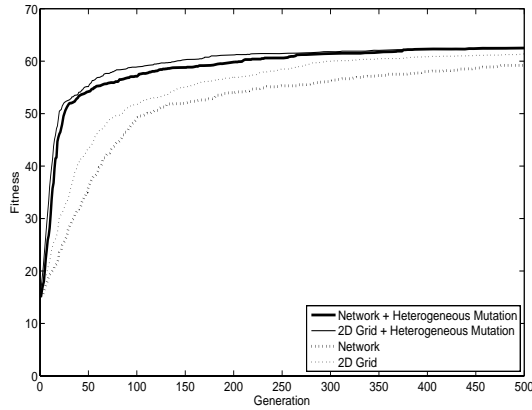


Figure 6: Comparison of convergence property.

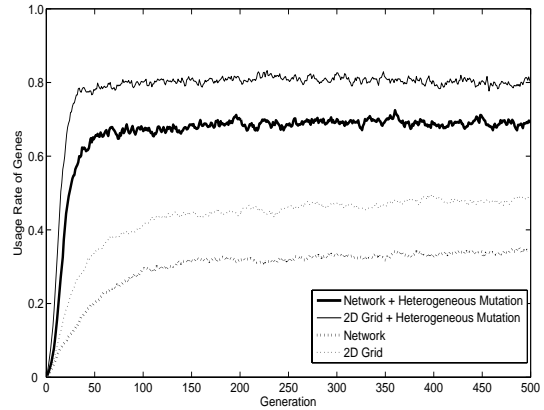


Figure 8: Comparison of the transition of the gene usage rate.

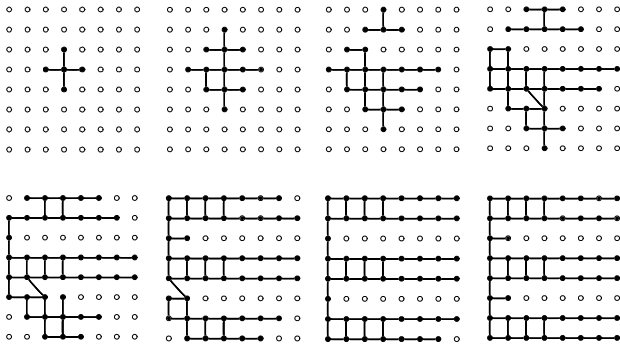


Figure 7: An example of the developmental process of “E” with our AE model.

Furthermore, an additional action is involved in a network generating AE, namely “reconnection”. These differences in operator implementation can be considered to make the task of generating predefined target patterns more difficult. Nonetheless, the result shows the evolvability of our AE model to be almost equivalent to the 2D grid AE with heterogeneous mutation.

Heterogeneous mutation is shown to be efficient in significantly improving evolvability for the evolution of both 2D grid AEs and network generating AEs. Activated genes are preserved at a rate of 0.99, while those inactive are randomized at the rate of 1.0. In other words, each gene can be regarded as under pressure to be activated. With this in mind, heterogeneous mutation reduces the number of inactive genes that make no contribution whatsoever to the objective function. Note that activated genes do not always contribute to the objective function. Subsequently, the reduction of such unnecessary genes improves the ability to explore a search space, i.e. the enhancement of evolvability.

Figure 8 shows the transition of the gene usage rate of the four types of model. Let n_a be the number of activated genes and n_e be the number of existing genes in a whole population. The usage rate of genes u_r is defined as follows:

$$u_r = n_a / n_e$$

The result shows that heterogeneous mutation improves the gene usage rate of the populations as well as evolvability.

5. EVOLUTION OF A ROBOT WITH A NETWORK STRUCTURE

We show that our AE model can be applied to the evolution of a robot with a network structure. We chose a software package called “Sodaplay” as a robot simulator.

5.1 Target Problem

Sodaplay is a 2D physical simulator developed by Soda Creative Ltd. in England [13]. Sodaplay software simulates physical environmental characteristics within the 2D space, such as gravity, surface friction, air friction and surface reflection. Users can construct their own Sodaplay robot that moves in complex motion by connecting mass points, springs and muscles. A muscle is equivalent to an oscillating spring.

The task is to generate a Sodaplay robot which traverses the pitfall-like terrain and reaches the goal as fast as possible (Figure 9). The value e_v that the AE program directly receives from Sodaplay software in order to evaluate each individual is defined as follows:

$$e_v = \begin{cases} t_c \times d_{sg} / d_a & (d_a > 0) \\ C & (d_a \leq 0) \end{cases}$$

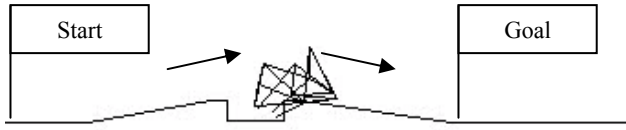


Figure 9: A Sodaplay robot with a network structure consisting of masses, springs and muscles.

Table 4: Elements of an action part for Sodaplay robots.

	Description	Value range
a	Connect, disconnect, or reconnect	0, 1, ..., 2*k
t	Type of a link	0, 1
l	Static length of a spring or muscle	1, 2, ..., 10
p	Phase of muscle oscillation	1, 2, ..., 10

Table 5: Parameters for the evolution of Sodaplay robots.

Population size	32
Number of elites	2
Number of genes	20
Number of iterations for growth	16
k (defining a neighborhood)	6
Necessary number of preconditions to be matched for a rule to be fired	6
Selection method	Roulette
Crossover method	Single point
Probability of crossover	1.0
Probability of mutation for activated rules	0.01
Probability of mutation for inactivated rules	1.0
Division number for phenotypic space	10*10
Number of masses (nodes)	16
Incest prevention threshold	80%

where t_e is the elapsed time for a robot to go ($t_e < t_{max}$: t_{max} is the timeout value). d_{sg} is the distance between the start and goal and d_a is the distance achieved toward the goal by the robot, while C is a large value arbitrary set. This definition provides a means of evaluation, whereby the value e_v diminishes the faster the robot reaches the goal. Subsequently, the fitness value of each individual is derived from the inverse value of e_v .

We add three elements in an action part because a Sodaplay robot includes two types of links, namely a spring and a muscle (Table 4). The element “t” defines the type of generated link while the element “l” defines the static length of the spring or muscle and the element “p” defines the muscle oscillation phase. Reconnection is applied only when the type of two links is identical. The type of spring or muscle newly generated by reconnection is the same as the type of the two links, while the other attributes of the new link are given by the loci of the gene. Note that these additional parts are not necessary if we simply generate network structures.

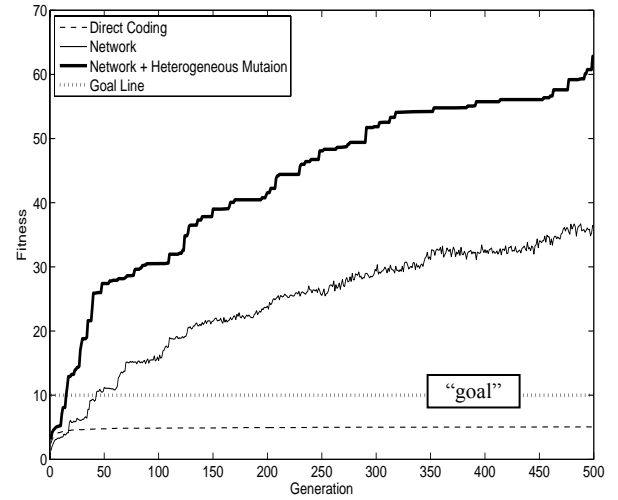


Figure 10: The convergence properties of the network generating AEs and the direct coding method.

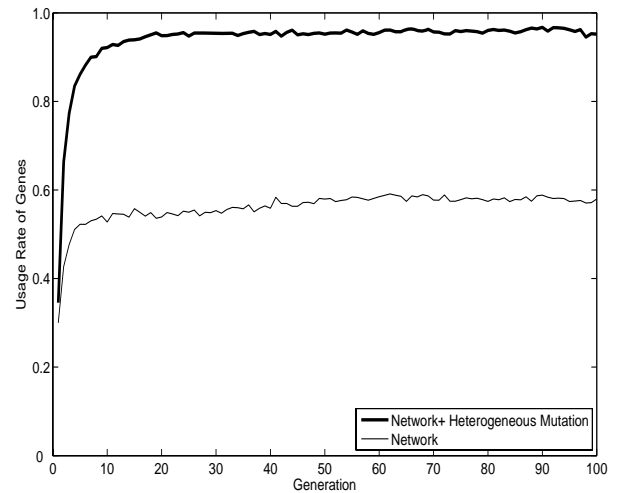


Figure 11: The transition of the gene usage rate of the network generating AEs.

5.2 Experimental setting

We evolved a Sodaplay robot with the network generating AE with heterogeneous mutation, setting its parameters as shown in Table 5. For comparison of its evolvability, we also used the direct coding method. In the direct coding method, we evolved an adjacent matrix and corresponding matrices describing the attributes of the links. As for incest prevention, two parents were reselected by roulette selector if more than 80% of the loci of the two parents were the same. This mechanism was expected to be effective in avoiding premature convergence.

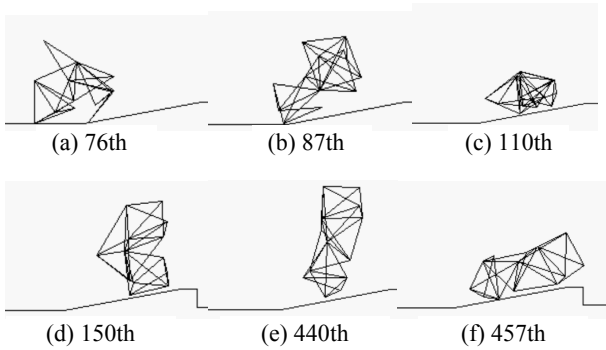


Figure 12: An individual with maximum fitness in each generation.

5.3 Results

Figure 10 shows the convergence property of network generating AE with heterogeneous mutation compared to the case without heterogeneous mutation and the direct coding method. We emphasize that conventional IE cannot be applied to the generation of this Sodaplay robot. We averaged the results of 20 runs of 500 generations. The horizontal line “goal” in Figure 10 shows the minimum fitness value required for reaching the goal. The result shows that heterogeneous mutation significantly improves evolvability. Also note that the robot generated by the direct coding method cannot even reach the goal.

Figure 11 shows the transition of the gene usage rate comparing the case both with and without heterogeneous mutation for the network generating AE. The gene usage rate of the case with heterogeneous mutation converges far more swiftly to about 97% than that without heterogeneous mutation, which converges to about 56%. Heterogeneous mutation is found to improve the evolvability as well as the gene usage rate in evolving a Sodaplay robot.

Figure 12 shows examples of robots having appeared over the course of evolution, all of which are individuals with maximum fitness in each generation. The robots shown in Figure 12 (e) (f) use a bend-and-stretch motion to leap forward and overcome the pitfall. It is also notable that a hind-leg-like structure used to hop forward was initially observed in the 76th generation. This immature robot is only capable of climbing up the slope and cannot overcome the pitfall. However, this hind-leg-like structure is inherited through generations, and finally becomes the key in leaping over the pitfall.

Figure 13 shows the developmental process of the robot shown in Figure 12 (f). The dashed line represents the spring and the solid line represents the muscle, with these figures only showing the initial states for the Sodaplay simulator’s run. It is interesting that the characteristic of autonomous size keeping has emerged through the evolutionary process. No constraint condition was given for the developmental process except the fixed number of 16 iterations. Nonetheless, the developmental process of the evolved robot is completed in no more than 5 iterations, and so 13 of the given 16 nodes are used to construct the robot.

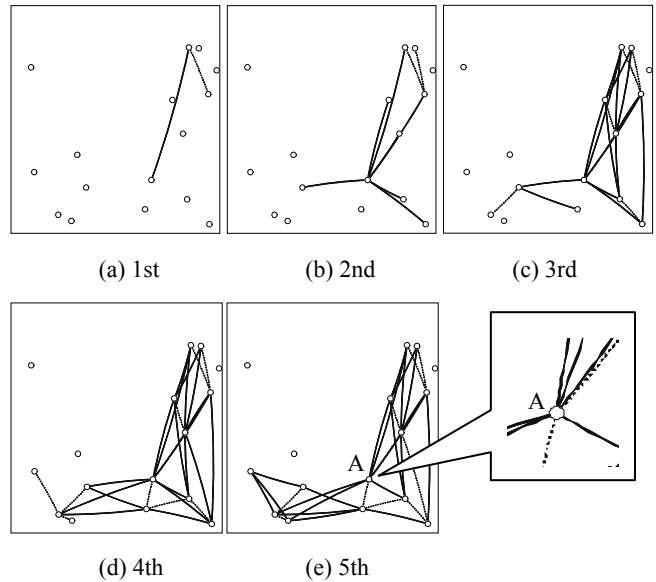


Figure 13: The developmental process of the individual with maximum fitness in the 457th generation.

It is also notable that the node “A” in Figure 13 has 8 links. This node plays a role as the center of the body within the physical space of Sodaplay software and successfully supports the bend-and-stretch motion, enabling jumping forward. This deviation of the neighborhood relationship is attained by the reconnecting action. Without reconnection each node can have only a maximum of 6 links, for the number $k=6$ limits the maximum number of locally available links.

6. DISCUSSION

Using conventional AE models for generating network structures out of local interactions, the generated structures are also inclined to be biased by the local neighborhood relationship. This means that solutions are searched in a highly reduced space, which represents only a portion of all possible combinations. In contrast, the structures generated by our novel AE model are less inclined to be biased by the homogeneous local neighborhood structures of the lattice. This is because the phenotypes are directly generated as network structures through the developmental process. Our model can search a larger space and generate long links, unrestricted by a neighborhood relationship.

Lucas et al. evolved Sodaplay robots using planar graph encoding [10]. In their work, the minimum element for development is a triangular structure consisting of three masses and three springs or muscles. With this approach, only a limited space can be searched.

Although few works exist resembling our own, probably the closest is “genobots” by Hornby et al. [5]. Genobots are robots with network structures. They are evolved using an L-system to describe the developmental process and complex motions can be achieved by employing neural controllers. However, the experiments of Genobots are conducted for the evolution of faster movement only on the plane terrain.

7. CONCLUSIONS AND FURTHER WORK

In this paper, we presented a novel network generating AE model involving reconnection and a heterogeneous mutation model. We showed that reconnection is efficient for generating a link to deviate from predefined local neighborhood relationship. The heterogeneous mutation model was shown to be significantly effective in improving its evolvability and enabled a larger solution space to be searched. We showed that our AE model can efficiently evolve network structures.

As an example task to be solved, we evolved a Sodaplay robot, which can be defined as a network structure. The developmental process of an evolved robot showed a size keeping property. The springs or muscles generated between nodes were in a relationship that transcended the predefined neighborhood. The evolved robot hopped forward and overpassed the pitfall-like terrain.

Our model can also evolve other network structures with simple modification. One direction of future work would be to apply the model for the evolution of truss structures or ANNs.

The heterogeneous mutation mechanism involved in our AE model can be applied to all kinds of rule-based evolutionary algorithm. Another direction of future work would be to experiment with the effectiveness in other rule-based evolutionary algorithms such as Learning Classifiers Systems.

8. REFERENCES

- [1] P. J. Bentley and S. Kumar. Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-1999*, pages 35–43. Morgan Kaufmann Publishers, July 1999.
- [2] P. Eggenberger. Evolving morphogenesis of simulated 3d organisms based on differential gene expression. In *Proceedings of the Fourth European Conference on Artificial Life*, pages 205–213. MIT Press, July 1997.
- [3] D. Federici. Evolving a neurocontroller through a process of embryogeny. In *Proceedings of the 8th International Conference on the Simulation of Adaptive Behavior: From Animals To Animats, 8*, pages 373–382. MIT Press, July 2004.
- [4] S. Harding and J. Miller. The dead state: A comparison between direct and developmental encodings. In *Genetic and Evolutionary Computation Conference (GECCO-2006) Workshop Program: Complexity through Development and Self-Organizing Representations (CODESOAR)*, ACM, July 2006.
- [5] G. S. Horby and J. B. Pollack. Creating high-level components with a generative representation for body-brain evolution. *Artificial Life*, 8(3), 223–246. 2002.
- [6] R. Kicingier. Generative design in civil engineering using cellular automata. In *New Kind of Science 2006 Wolfram Science Conference*. June 2006.
- [7] T. Kowaliw, P. Grogono, and N. Kharma. The evolution of structural design through artificial embryogeny. In *Proceedings of IEEE First International Symposium on Artificial Life*, pages 425–432. IEEE, April 2007.
- [8] T. Kowaliw, P. Grogono, and N. Kharma. Environment as a spatial constraint on the growth of structural form. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2007*, pages 1037–1044. ACM, July 2007.
- [9] S. Kumar and P. J. Bentley. The ABCs of evolutionary design: Investigating the evolvability of embryogenies for morphogenesis. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-1999*, pages 164–170. Morgan Kaufmann Publishers, July 1999.
- [10] S. Lucas. Evolving spring-mass models: test-bed for graph encoding schemes. In *Proceedings of IEEE Congress on Evolutionary Computation, CEC 2002*. pages 1952–1957. IEEE, May 2002.
- [11] J. F. Miller. Evolving a self-repairing, self-regulating, French flag organism. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2004*, pages 129–139. Springer Verlag, June 2004.
- [12] J. F. Miller and Peter Thompson. Beyond the complexity ceiling: Evolution, Emergence and Regeneration. In *Genetic and Evolutionary Computation Conference Workshop on Regeneration and Learning in Developmental Systems, GECCO-2004*, Springer Verlag, June 2004.
- [13] Soda Creative Ltd. Sodaplay homepage. <http://sodaplay.com/>
- [14] K. O. Stanley. Comparing Phenotypes with Natural Biological Patterns. In *Genetic and Evolutionary Computation Conference (GECCO-2006) Workshop Program: Complexity through Development and Self-Organizing Representations (CODESOAR)*, ACM, July 2006.
- [15] K. O. Stanley and R. Miikkulainen. A taxonomy for artificial embryogeny. *Artificial Life*, 9(2): 93–130. April 2003.
- [16] S. Uyar, S. Sariel and G. Eryigit. A gene based adaptive strategy for genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2004*, pages 271–281. Springer Verlag, June 2004.
- [17] S. Viswanathan and J. Pollack. How artificial ontogenies can retard evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2005*, pages 273–280. ACM, June 2005.