Development, Learning and Evolution in Animats

Jérôme Kodjabachian and Jean-Arcady Meyer

Ecole Normale Supérieure
Groupe de BioInformatique - URA 686
75230 Paris cedex 05, France
E-mail: kodjaba@wotan.ens.fr, meyer@wotan.ens.fr

Abstract

This paper successively describes the works of Boers and Kuiper, Vaario, Nolfi and Parisi, Gruau, and Dellaert and Beer, which all evolve the developmental program of an artificial nervous system. The potentialities of these approaches for automatically devising a control architecture linking the perceptions and the actions of an animat are then discussed, together with their possible contributions to the fundamental issue of assessing the adaptive values of development, learning and evolution.

Keywords

Development – Learning – Evolution – Animats – Sensorimotor Control

1 Introduction

An animat [MEYE91a, MEYE92, CLIF94] is an artificial organism - either a simulated animal or an animal-like robot - the structure and functionalities of which are based substantially on mechanisms observed in real animals. It is usually equipped with sensors, with actuators, and with a behavioral control architecture that relates its perceptions to its actions and allows it to survive in its environment.

Such a control architecture can be fixed by a human designer or by an automatic process inspired from biology and involving the three main adaptive processes characteristic of living systems, i.e. the processes of development, learning and evolution.

Although learning and evolution have already often been used for the automatic design of control architectures in animats [MEYE91b, MEYE94], such does not happen to be the case with the process of development, a point stressed and regretted by Meyer and Guillot [MEYE94]. However, a few such applications - which combine development, evolution and, possibly, learning - have recently been published. They will be described in the remainder of this paper, which will close with a discussion of the foreseeable potentialities of such approaches.

2 Boers and Kuiper

The work of Boers and Kuiper [BOER92] combines a genetic algorithm and a learning procedure with a Lindenmayer grammar [LIND68] that models development. Basically, the genetic information on which the genetic algorithm operates codes for a set of production rules which are applied to an axiom for a number of iterations. The resulting string is transformed into a structural specification for a classical feed-forward neural network. The weights of this network are trained by back-propagation, which provides a fitness estimate that is returned to the genetic algorithm.

The strings used in this work are made up of 16 characters from the alphabet {A-H, 1-5, [,]} U {,}.
A letter (A-H) designates a specific neuron in the network and two adjoining letters are automatically connected feedforward. If two letters are separated by a comma (,), no connection is made. Modules can be created by grouping neurons or other modules between square brackets ([,]): two adjoining modules are connected - so that all output neurons from the first module are connected to all input neurons from the second module - and two modules separated by a comma are not connected.

Single digits are used to denote a skip within the string. For instance, the string [A2[B,C]D]E codes for the network of Figure 1, where neuron A is connected to neurons B and C because both are input neurons in the [B,C] module. Neuron A is also connected to neuron E because the connection skips both module [B,C] and neuron D.

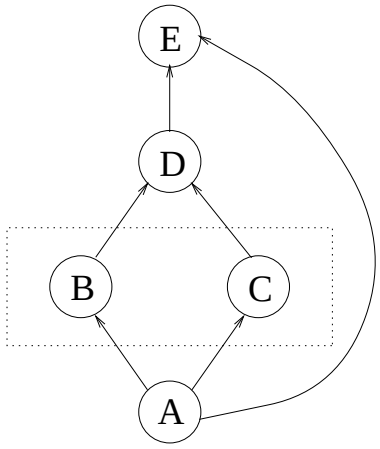

Figure 1. The string [A2[B,C]D]E developed. After [BOER92].

1 : A → BBB
2 : B > B → [C, D]
3 : B → C
4 : C < D → C
5 : D > D → C1

Figure 2. A sample of production rules. After [BOER92].

The L-system used for generating such strings is a 2L-system, in which every production rule can have both left and right contexts and is therefore divided into four (possibly empty) parts: $L < P > R \rightarrow S$.
Basically, such a rule means that sub-string P (the predecessor) should be replaced by sub-string S (the successor) if P is connected to every neuron described in L (left- or lower-level context) and in R (right- or upper-level context). Thus, if the five production rules of Figure 2 are applied to a single original neuron A - given as an axiom - (Figure 3a), the string BBB is generated after one rewriting step (Figure 3b). During the second rewriting step, the first (bottom) and the second (middle) B's are rewritten using rule 2, because they both are connected to a higher B. On the contrary, the third (top) B - which has no connection with a higher B - is rewritten according to rule 3, instead of rule 2 (Figure 3c). Likewise, during the third rewriting step, the first (bottom) D is rewritten according to rule 5 and the second (middle) according to rule 4. As no more rules apply, the final network obtained corresponds to string [C, C1] [C,C] C shown in Figure 3d.

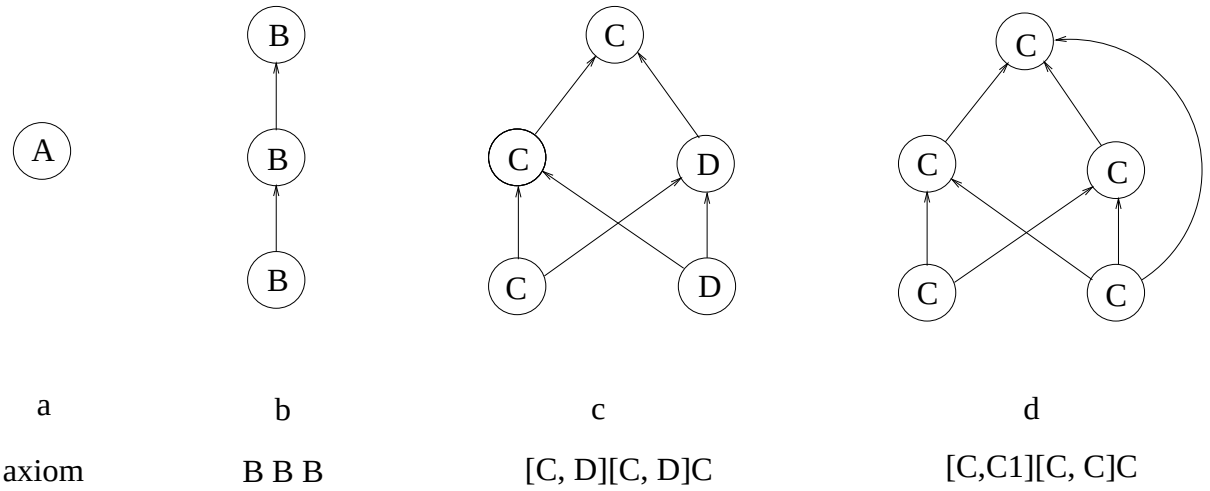

Figure 3. The development of a neural network using rules of Figure 2. After [BOER92].

To separate the constituent parts of each production rule, Boers and Kuiper use a special character (an asterisk) and, to relate each of the 17 possible characters in a production rule to the genetic information processed by the genetic algorithm, they use the genetic code described in Figure 4. Thus, in this application, the genetic code relates 17 characters to 64 6-bit strings, instead of relating 20 amino-acids to 64 triples with 4 bases.

	00	01	10	11	
00	3	[	D	]	00
	3	[	D	]	01
	*	[	2	2	10
	*	[	2	5	11
01	*	1	E	]	00
	*	1	E	]	01
	*	1	F	]	10
	*	1	F	]	11
10	2	A	G	[	00
	2	A	G	[	01
	2	A	H	]	10
	4	A	H	]	11
11	,	B	*	C	00
	,	B	*	C	01
	,	B	[	C	10
	,	B	[	C	11

Figure 4. The genetic code used in [BOER92]. For example, the character corresponding to string 100100 is the first A in the Table.

Furthermore, the genetic information on a given chromosome can be read in twelve different ways - starting at any of the first 6 bits and reading forward, or starting at any of the last 6 bits and reading backwards - thus providing the genetic algorithm with a much higher level of implicit parallelism than in traditional applications. Figure 5, for instance, describes 4 different

translations of a chromosome with a length of 48.
Finally, the software developped by Boers and Kuiper also contains several functions capable of repairing faulty strings, i.e. strings with extraneous brackets, useless commas, or succeeding digits.
To our knowledge, this software has so far been used in only a few very simple applications. For instance, it has evolved neural networks capable of solving the XOR problem or of recognizing handwritten digits 0,1 ... 9 presented on a 5x5 grid.

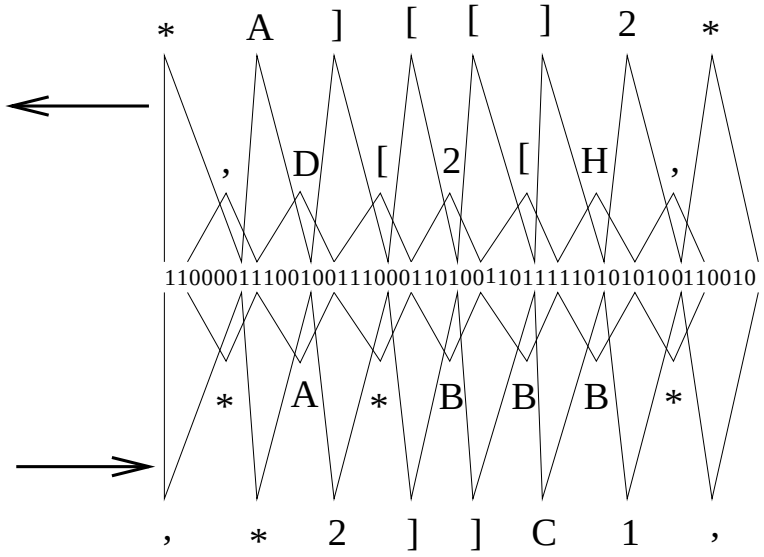

Figure 5. Extract from a chromosome together with 4 possible translations. After [BOER92]

3 Vaario

Vaario's approach [VAAR93, VAAR94] explicitly takes into account environmental effects on the development of neural networks and is also inspired by Lindenmayer's systems [LIND68]. However, instead of using linear character strings, it makes use of abstract objects which typically represent artificial cells - each characterized by a set of attributes and a set of production rules to execute. In this model, each cell is actively "checking" the environment and, on the basis of the corresponding information, executes one or several of its production rules. Cell attributes mostly refer to the concentrations of various chemical elements and enzymes. Production rules are characterized by the set of conditions which must be fulfilled for them to be executable and by the kind of action they trigger. They are divided in 4 types:
-*cytoplasm rules*, interpreting the genetic code and modifying the internal state of a cell;
-*membrane rules*, modifying the internal state of a cell according to the interactions between the cell and its environment;
-rules creating a cell;
-rules deleting a cell.
In particular, these rules are used to model various morphogenetic processes, such as cell division, axon and dendrite growth, axon guidance and target recognition, cell death, elimination of connections, anatomical plasticity and synaptical plasticity (Figure 6).

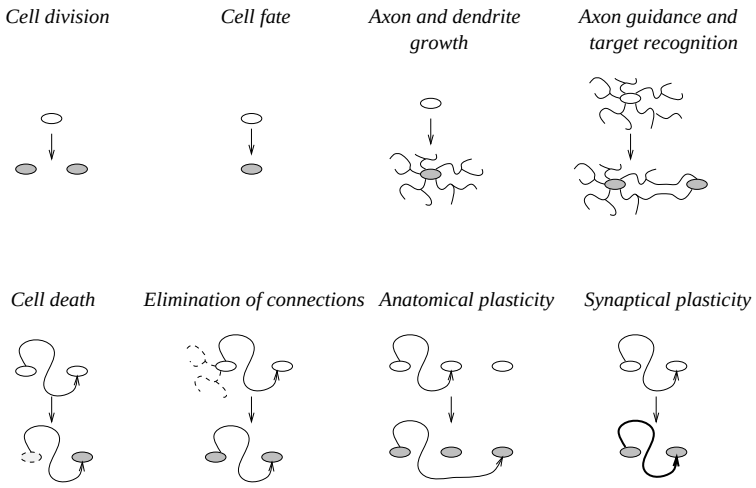

Figure 6. Some morphogenetic processes. After [VAAR93].

For example, the process of axon and dendrite growth depends on the presence of obstacles and of target cells in the environment. Connections bounce against obstacles and climb the gradient fields of the chemical substances emitted by target cells. When a connection finally reaches a target cell, it creates a synaptic connection and stops growing. Moreover, those connections unable to find any target neuron gradually withdraw.

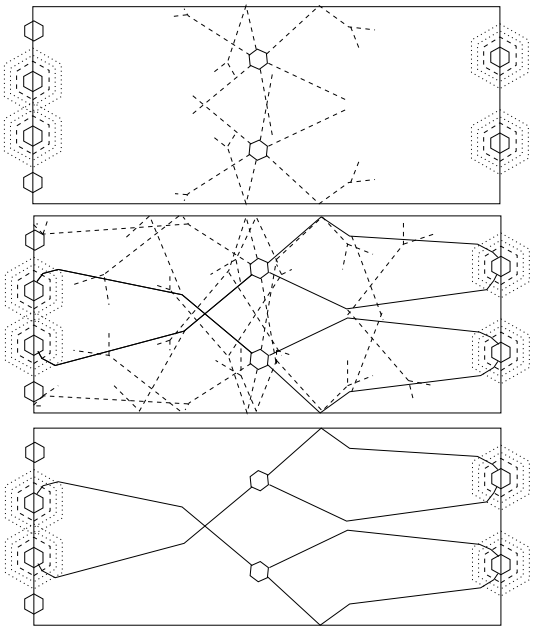

Figure 7. Three developmental stages in the development of Vaario's animat. Three phases are shown: initial growth (top), initial withdrawal (middle) and after all unconnected links are withdrawn (bottom). After [VAAR94].

In the current implementation of Vaario's model, the genotype of an animat is not encoded in a bit string, but in a symbolic representation which also allows crossover and mutation operations. Thus, several characteristics - like the time to branch, the branching angle and the

type of target cells involved in connection growth, or the numbers, positions and properties of the animat's sensors and actuators - are genetically determined.
Figure 7 shows the development of the nervous system of an animat with two sensors and four actuators. The cell positions and the targeting labels (i.e. which neuron will be connected to which sensors and actuators) have been given explicitly. Figure 8 shows what kind of neural network can be evolved in order to generate a tracking behavior. The signal generated by each sensor is a genetically coded function of distance and angle of the stimulus. Likewise, each actuator generates a force which depends on the incoming signal in a genetically determined manner. Although the model doesn't incorporate any learning ability in its present stage of development, it wouldn't be difficult to allow some production rules to modify thresholds or connection weights in future implementations.

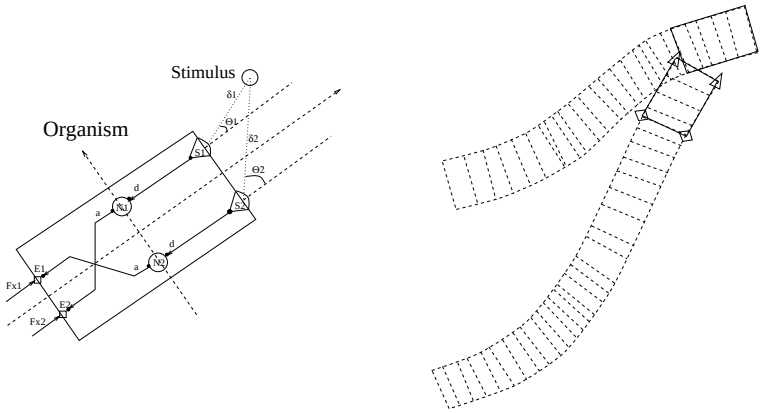

Figure 8. The developed nervous system and the tracking behavior of Vaario's animat. After [VAAR94].

4 Nolfi and Parisi

The work of Nolfi and Parisi [NOLF91] is concerned with the evolution of animats that can consume food elements, randomly distributed in the environment. Each animat is equipped with a sensory system that allows it to perceive the direction and the distance of the nearest food element and with a motor system that provides the possibility of turning any angle between 90 degrees left and 90 degrees right, and to move forward 0 to 5 steps. The nervous system of each animat is a bidimensional network with up to 40 neurons, whose development is coded in the animat's genotype. This genotype is a fixed-length string of 40 blocks, each block being made up of eight genes that describe the developmental fate of a given neuron. The first five blocks in the string correspond to sensory neurons, the last five blocks to motor neurons and the 30 intermediate blocks to internal neurons, which can be arranged in a maximum of 7 layers.
Within a given block, the first gene is a *temporal expression gene* which specifies when during development the corresponding neuron will be expressed. Neurons scheduled to appear after the animat's death are non-expressed neurons.
Two *physical-position genes* represent respectively the x and y spatial coordinates of the corresponding neuron.
The *branching-angle gene* and the *segment-length gene* respectively control the angle of each branching of the neuron's axon and the length of each branching segment.

The *synaptic-weight gene* determines the synaptic weight of each connection established by the corresponding neuron. In other words, in this model, all connections originating in a given neuron have the same weight.
The *bias gene* represents the activation bias of the corresponding neuron.
Lastly, the *neuron-type gene* specifies, in the case of a sensory neuron, whether this neuron reacts to the angle or the distance of food and, in the case of a motor neuron, whether this neuron determines the angle of turn or the length of a forward step.
According to the developmental instructions coded in the genotype, the nervous system of each animat changes during the animat's life: some neurons are created at birth, others appear later, and connections are established between two neurons when the growing axonal branch of a particular neuron reaches the soma of another one.

Results obtained by Nolfi and Parisi suggest that the coupling of an evolutionary process and a developmental process allows the discovery of neural architectures enabling an animat to move in its environment and to capture food. Results also suggest that the architectures evolved tend to be structured in functional sub-networks.

In a recent extension of this work [NOLF94], both the genes and the environment influence the neural development because a neuron is allowed to grow its branching axon only if the neuron's activation variability - which depends upon the variability of the environmental stimulations to the network - exceeds a genetically specified threshold.

5 Gruau

The work of Gruau [GRUA92, GRUA93], like that of Boers and Kuiper, encodes a rewriting grammar in a chromosome. However, this encoding scheme - called *cellular encoding* - rewrites neurons instead of characters. In its simplest version, it is used to develop feedforward networks of Boolean neurons with integer thresholds and +1 or -1 connections, but more elaborated versions of this encoding scheme [PRAT94] can deal with more complex neurons and connectivities.

In Gruau's model, each cell in a developing network has a copy of the chromosome that codes the developmental process, and each cell reads the chromosome at a different position. The chromosome is represented as a grammar tree with ordered branches whose nodes are labeled with character symbols. These character symbols represent instructions for cell development that act on the cell or on connections that fan-in to the cell. During a step of the development process, a cell executes the instruction referenced by the symbol it reads and moves its reading head down in the tree. Depending on what it reads, a cell can divide, change some interval registers and finally become a neuron. For instance, when a cell reads and executes the *sequential division* (denoted by S), it divides into two linked cells: the first child inherits the input links, the second child inherits the output links of the parent cell. When a *parallel division* (denoted by P) is executed, both child cells inherit the input and output links from the parent cell. Since a given cell gives two child cells, S and P nodes are of arity two: the first child moves its reading head to the chromosome's left subtree and the second child moves its head to the right subtree. Finally, when a cell divides, the values of the internal attributes of the parent cell are copied in the child cells.

Other symbols change the values of internal registers in the cell. Some registers are used

during development - like the *link register* for instance, which points to a specific fan-in link or connection into a cell - while others determine the weights and thresholds of the final neural network. Thus, symbols I and D respectively increment and decrement the value of the link register, causing it to point to a different connection. Likewise, symbols A and O respectively increment and decrement activation thresholds, and symbols + and - respectively set to +1 and -1 the weight of the input link pointed by the link register. The *ending program* symbol E causes a cell to lose its reading head and become a neuron.

Figure 9 represents the development of a XOR network. Circles represent active cells or neurons, while rectangles represent reading heads. Empty circles correspond to thresholds set to 0, black circles correspond to thresholds set to 1. Squares represent input/output pointer cells. Continuous connections have a weight of 1, dashed connections have a weight of -1.

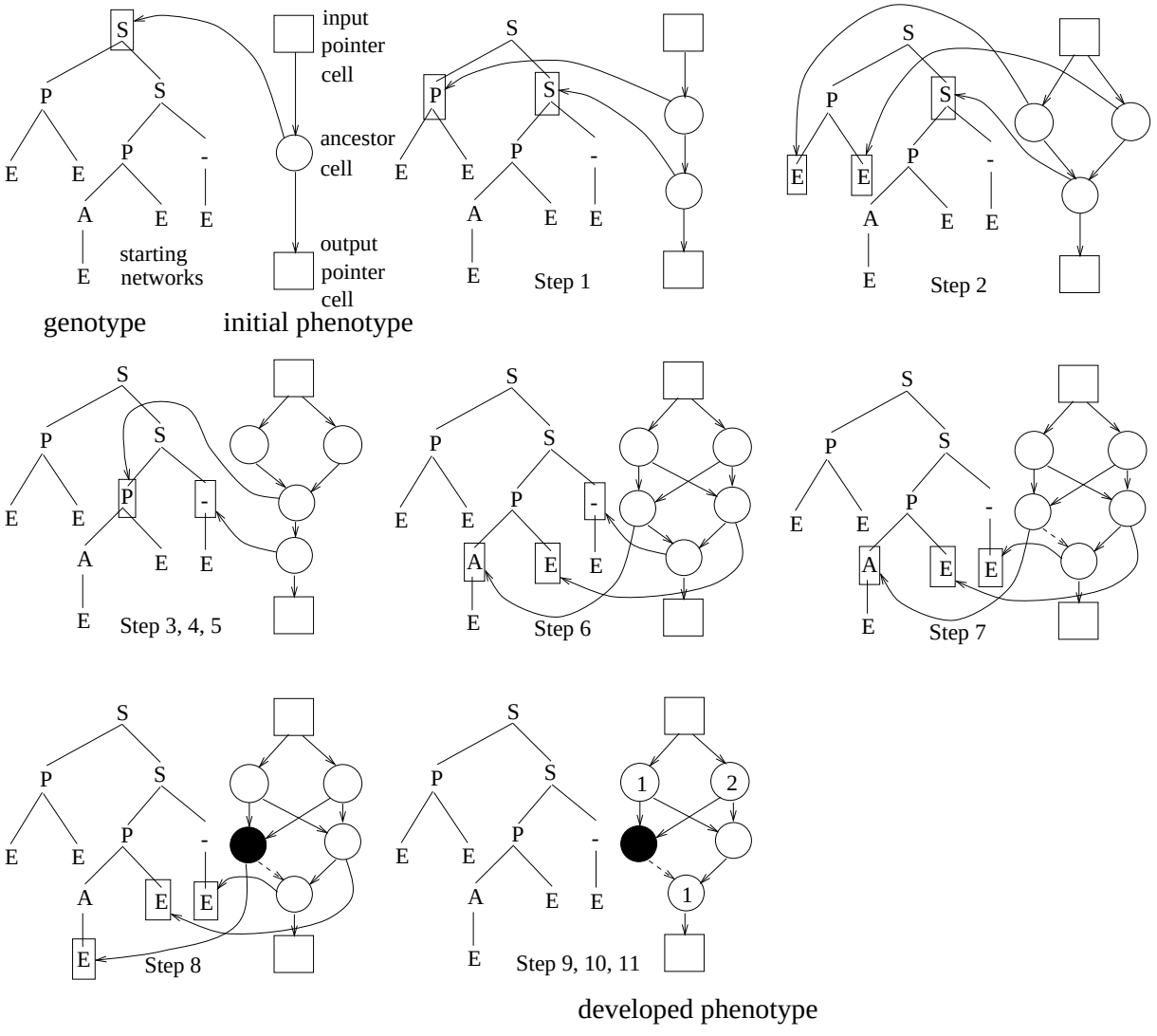

Figure 9. Cellular encoding and development of a XOR network. After [GRUA92].

Since Gruau's chromosomes have the same structure as those used by Koza within the Ge-

netic Programming paradigm [KOZA92], they can be subjected to the same kind of genetic operators, notably to mutations and crossing-overs.

Cellular encoding has been used by Gruau [GRUA94] to evolve a neural network capable of controlling the locomotion of a six-legged animat. This problem has already been solved by Beer and Gallagher [BEER93], who, instead of forming a locomotion controller by fully interconnecting six individual leg-controllers, took advantage of the various symmetries that such a controller was supposed to exhibit and devised a controller made of six copies of the same sub-network. Gruau solved a slightly simpler version of the problem, but did not help the evolutionary algorithm by using any a priori knowledge about symmetries. Instead, symmetries were discovered and exploited by the developmental process, because such a process can be capable of generating a sub-network that solves a sub-problem, then, of producing and combining copies of this sub-network to build a higher-level network that solves the problem. The genome splicing technique advocated by Koza [KOZA94] seems especially useful for such a purpose.

Gruau and Whitley [GRUA93] have added a variety of Hebbian learning to cellular development and evolution. In particular, following Hinton and Nowlan [HINT87] and Belew [BELE89], they have compared results obtained with fitness evaluations depending on a developed neural network alone to results obtained with fitness evaluation depending on a developed neural network with some of its weights changed by a learning procedure. It thus appears that such a modification changes the fitness landscape explored by the genetic algorithm and, eventually, accelerates the speed of evolution - a result known as the *Baldwin effect*. Likewise, Gruau and Whitley have studied how the so-called *developmental learning* could affect evolution. Such learning can occur when some recursive encoding is used by the cellular encoding method, thus allowing a given subtree of the chromosome to be repeatedly read and executed. In such circumstances, indeed, it is possible to learn and change the weight of a connection between two iterations of the recursive loop.

However, it should be stressed that neither the Baldwin effect, nor the developmental learning, pass the values of learned weights from parents to offspring and, thus, that they do not implement any Lamarckian inheritance of acquired characters.

6 Dellaert and Beer

The developmental model of Dellaert and Beer [DELL94] is inspired from Kauffman's work [KAUF69] and relies upon a genetic regulatory network whose binary elements each correspond to the presence (or absence) of a specific gene product or to the expression (or the non-expression) of some gene.

According to the updating rule and connectivity of each element, the state of the network - which corresponds to the pattern of gene expression in a given cell - may change over time but will, ultimately, settle in a fixed point or a limit cycle. Such a dynamic process is used to model a cell cycle: in particular, a cell division occurs when the cell's regulatory network settles in a steady state, with a specific element being set to a predetermined value. When this occurs, the pattern of gene expression of the parent cell is passed to the next generation, and a subset of genetic elements is used to determine the final differenciation of the two daughter cells.

Within such a framework, the morphology of an animat is a two-dimensional square consisting of cells of various types, each having a copy of the same Boolean network that constitutes the animat's genotype. However, the state of the network, corresponding to the pattern of gene expression in a particular cell, may be different in each cell, according to the cell's initial state and to the various influences experienced up to the present time.

The physical extent of each cell is represented as a two-dimensional square element that can divide in either of two directions, vertical or horizontal. When division occurs, it takes place in such a way that the longest dimension of the parent cell is halved and that the two daughter cells together take up the same space as the original cell.

Development starts out with one simple square that represents the zygote. During development, the state of the regulatory network of each cell changes according to both the internal dynamic process mentioned above and the external influences provided by intercellular communications or by specific symmetry-breaking processes.

For instance, the influence of neighboring cells is condensed into a so-called *neighborhood vector*, which is the logical OR of all the state vectors of these cells, and this neighborhood vector is combined with the cell's state vector to determine the next state. Likewise, a symmetry-breaking process occurs at the time of first cleavage, which switches a bit of the Boolean network state vector into one of the zygote's two daughter cells. Other symmetry-breaking processes cause the update of a cell's state vector to depend upon information on whether the cell is situated on the external surface of the animat or whether it borders the animat's horizontal midline.

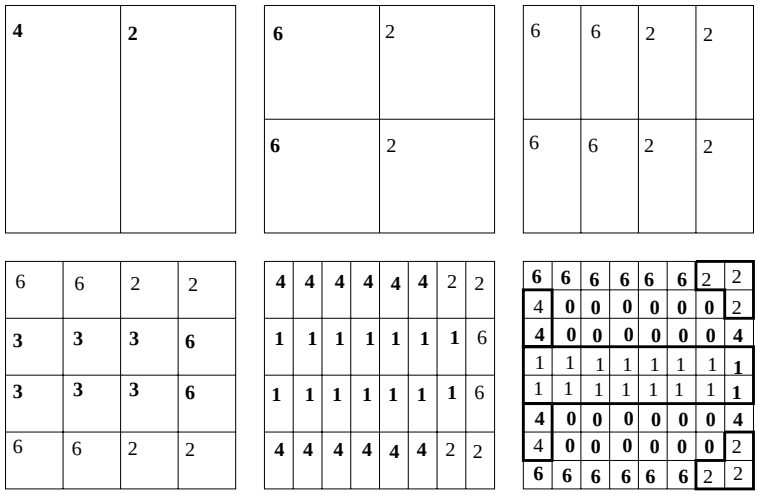

Figure 10. The first six consecutive stages of development of Dellaert and Beer's animat. After [DELL94].

Dellaert and Beer have evolved a simple animat that roughly reproduces the relative placement of sensors, actuators and control system that one would expect to find in a simple chemotactic agent (Figure 10). In particular, this animat exhibits bilateral symmetry, with sensors (cell type 2) placed sideways at the front and actuators (cell type 4) placed sideways at the

back, and with a control structure made of "neural tissue" (cell type 1) connecting them. Such an organization has been obtained by making the evolutionary process depend upon a fitness function that evaluates the discrepancies between the differenciation patterns of any developed animat and that of an hypothetical ideal chemotactic agent. In the case of the animat on Figure 10, these discrepancies have not been entirely eliminated, because two actuator cells are clearly out of place, in front of the animat.

a) node	0 1 0 1 0 0 1 1	inputs
1	0 0 1 0	3 -6
2	1 1 0 0	-2 -1
3	0 0 0 1	-5 5
4	1 1 0 1	4 4
5	0 1 1 0	6 -6
6	0 1 1 1	6 -1

b) node	Equivalent Boolean function
1	~3 AND mid
2	~(-1)
3	ext AND 5
4	~4 OR 4
5	6 XOR mid
6	6 OR -1

Figure 11. The genotype of Dellaert and Beer's animat. After [DELL94].

Figure 11 describes the genome of this animat, in two equivalent forms. It is a Boolean network with six nodes, each characterized by a specific update rule which sets the state of the corresponding node according to information contributed by two inputs. These inputs are also genetically determined and represent connections from other nodes (positive integers), connections from nodes in neighboring cells (negative integers in the range [-4, -1]) or influences of the external environment (-5) or of the midline (-6).

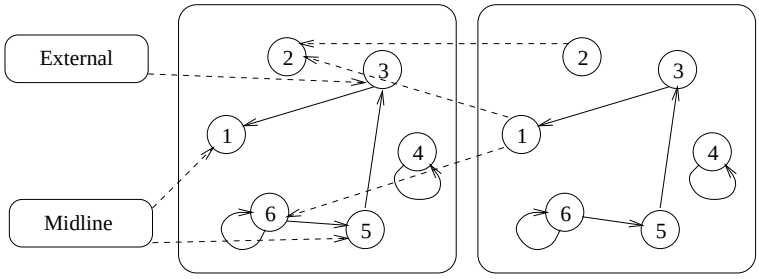

Figure 12. Intracellular and extracellular communications in Dellaert and Beer's animat. After [DELL94].

Thus, as shown in Figure 12, the update rule of node 1 in a given cell depends upon the state of node 3 in the same cell and upon the situation of this cell relative to the animat's midline: if the cell borders the midline, the value of bit -6 is 1, otherwise it is 0. Likewise, the update rule of node 3 depends upon the state of node 5 and upon the situation of this cell relative to the animat's external surface: if the cell is situated on this surface, the value of bit -5 is 1, otherwise it is 0. As Figure 12 also shows, the update rules of nodes 2 and 6 in a given cell

depend on the state of node 1 in neighboring cells or, more precisely, on the state of bit 1 in the cell's neighborhood vector.

In this application, cellular division was dependent upon the state of node 4, whose updating rule always responded 1 (because ~4 OR 4 is always TRUE) and maintained this node in a permanently active state. Thus, a division occurred at every step, resulting in a maximum number of cells.

7 Discussion

Although it is somewhat premature to speculate on the relative merits of such preliminary approaches, it is clear that, however different from each other they may be, they are all capable of developing the control architecture of an animat. Therefore, they should prove useful in the future, at least in a purely engineering perspective.
As compared to other evolutionary approaches that bypass the process of development and directly map the genotype into the phenotype (see reviews in HUSB94, MEYE91b, MEYE94, SCHA92), the use of a developmental model should not only be capable of generating more complex phenotypes with simpler genotypes, but also of exhibiting some properties whose adaptive values should be extremely valuable for animats' control. In particular, developmental processes are suitable for generating modular architectures, thus providing an animat with the important functionality of problem decomposition - as examplified by Gruau's work. Likewise, developmental processes easily provide symmetry-breakink mechanisms - as demonstrated by Dellaert and Beer's approach - whose effects on the resulting architecture are extremely difficult and tedious to code in a direct genotype-to-phenotype mapping.

The use of developmental models in conjunction with an evolutionary process should also prove to be valuable in the future in a more fundamental perspective. Indeed, such an approach obviously makes it possible to study how genetic information and environmental influences interact and complement each other during development. In particular, this approach should help in specifying for which environment and for solving which kind of survival problem, Nature has been committed to inventing the process of development. In other words, it should help in assessing the adaptive value of this process and in specifying how it interacts with those of learning and evolution.

8 Conclusions

This paper has described five recent approaches combining the three main adaptive processes exhibited by natural systems, i.e. those of development, learning and evolution. Although it is not yet possible to assess the relative merits of these approaches - which are quite different from each other - there are good reasons to think that they will prove helpful for automatically designing efficient control architectures linking perception to action in animats. These approaches should also provide a valuable contribution to theoretical biology and enable a better understanding of the interaction between development, learning and evolution to be gained.

References

[BEER92] R.B. Beer and J.C. Gallagher. Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior*. 1, 1:91-122. 1992.

[BELE89] R.K. Belew. When both individuals and populations search: Adding simple learning to the genetic algorithm. In *Proceedings of the Third International Conference on Genetic Algorithms*. D. Schaffer (Ed.). Morgan Kaufmann. 1989.

[BOER92] E.J.W. Boers and H. Kuiper.Biological metaphors and the design of modular artificial neural networks. *Master's Thesis. Leiden University, The Netherlands*. 1992.

[CLIF94] D. Cliff, P. Husbands, J.A. Meyer, and S.W. Wilson (Eds). *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*. MIT Press/Bradford Books. 1994.

[DELL94] F. Dellaert and R.D. Beer. Toward an evolvable model of development for autonomous agent synthesis. To appear in *Proceedings of the Artificial Life IV Conference*. 1994.

[GRUA92] F. Gruau. Genetic systems of boolean neural networks with a cell rewriting developmental process. In *Combination of genetic algorithms and neural networks*. D. Whitley and J.D. Schaffer (Eds). IEEE Computer Society Press. 1992.

[GRUA93] F. Gruau and D. Whitley. Adding learning to the cellular development of neural networks: Evolution and the Baldwin effect. *Evolutionary Computation*. 1, 3:213-234. 1993.

[GRUA94] F. Gruau. Efficient computer morphogenesis: A pictorial demonstration. *T.R. 94-04-027. Santa Fe Institute*. 1994.

[HINT87] G.E. Hinton and S.J. Nowlan. How learning can guide evolution. *Complex systems*. 1:495-502. 1987.

[HUSB94] P. Husbands, I. Harvey, D. Cliff and G. Miller. The use of genetic algorithms for the development of sensorimotor control systems.*Proceedings of the PerAc'94 Conference*. IEEE Computer Society Press. 1994.

[KAUF69] S. Kaufmann. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*. 22:437-467. 1969.

[KOZA92] J.R. Koza. *Genetic programming: A paradigm for genetically breeding computer population of computer programs to solve problems*. MIT Press. 1992.

[KOZA94] J.R. Koza. *Genetic programming II: Automatic discovery of reusable programs*. MIT Press. 1994.

[LIND68] A. Lindenmayer. Mathematical models for cellular interaction in development. Part I and II. *Journal of Theoretical Biology*, 18:280-315. 1968.

[MEYE91a] J.A. Meyer and S.W. Wilson (Eds). *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*. MIT Press/Bradford Books.1991.

[MEYE91b] J.A. Meyer and A. Guillot. Simulation of adaptive behavior in animats: Review and prospect. In [MEYE91a].

[MEYE92] J.A. Meyer, H.L. Roitblat, and S.W. Wilson (Eds). *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*. J.A. Meyer and S.W. Wilson (Eds). MIT Press/Bradford Books. 1993.

[MEYE94] J.A. Meyer and A. Guillot. From SAB90 to SAB94 : Four Years of Animat Research. In [CLIF94].

[NOLF91] S. Nolfi and D. Parisi. Growing neural networks. *T.R. PCIA-91-15. Institute of Psychology. Rome.* 1991.

[NOLF94] S. Nolfi, O. Miglino and D. Parisi. Phenotypic plasticity in evolving neural networks. *Proceedings of the PerAc'94 Conference*. IEEE Computer Society Press. 1994.

[PRAT94] P. Pratt. Evolving neural networks to control unstable dynamical systems. To appear in *Proceedings of the Evolutionary Programming'94 Conference*. 1994.

[SCHA92] J.D. Schaffer, D. Whitley and L.J. Eschelman. Combinations of genetic algorithms and neural networks: A survey of the state of the art. In *Combinations of genetic algorithms and neural networks*. D. Whitley and J.D. Schaffer (Eds.). IEEE Computer Society Press. 1992.

[VAAR92] J. Vaario. An emergent modeling method for artificial neural networks. *PhD Thesis. University of Tokyo*. 1993.

[VAAR94] J. Vaario. From evolutionary computation to computational evolution. To appear in *Informatica*. 1994.