

DIFFERENTIABLE CHROMOSOMES

The Genetic Programming of Switchable Shape-Genes

Hugo de GARIS ^{1,3,4} Hitoshi IBA ², Tatsumi FURUYA ¹

1) Computational Models Section, 2) Machine Inference Section,
Computer Science Division, Electrotechnical Laboratory (ETL),
1-1-4 Umezono, Tsukuba-shi, Ibaraki, 305 Japan,
email : DEGARIS@ETL.GO.JP

3) Center for Artificial Intelligence, George Mason University,
4) CADEPS Artificial Intelligence and Artificial Life Research Unit, Universite Libre de Bruxelles,

Keywords : Artificial Embryology, Differentiable Chromosomes, Genetic Programming, Genetic Algorithms, ALife, Operons, Gene Switching, Shape Genes, Reproducible Cellular Automata, Evolvability, Molecular Scale Technologies, Homogeneous and Heterogeneous (Avogadro) Machines, Molecular Electronics, Complexity, Neural Evolution, Darwin Machines, Embryonics (Embryological Electronics).

Abstract : This paper has three main aims. The first is rather general, and attempts to generate an awareness that there is a growing need to study the creation of self-assembling (i.e. embryonic like) systems, especially in the rapidly approaching age of molecular scale technologies. The remaining two aims are more specific. The second aim is to introduce into the traditional scope of Genetic Algorithms [GOLDBERG 1989] the computational equivalent of biological "differentiation", (i.e. where groups of genes (operons) switch on and off, producing different proteins and hence different phenotypes in an ordered, time sequential, "embryological" way). The third aim is to show how these ideas can be used to "grow" nonconvex artificial shapes (or "embryos") of colonies of cells within the framework of reproducible two-dimensional cellular automata. A GA chromosome is divided into regions called "operons", where each operon consists of a condition part and an action part. The combined influence of the action parts of the active (switched on) operons can change the states of cells. If the states match the condition parts sufficiently of other operons, then these operons can switch on and execute their action parts. This operon switching (or differentiation) of shape generating "genes" allows the growth of some nonconvex shapes. However, the growth of complex nonconvex shapes remains very difficult.

1. Introduction

The broad aim of this paper is to create an awareness that self assembling, "artificial embryological" type systems need to be studied, so that future molecular scale technologies [DREXLER 1992] will be able to build machines which have too many components for sequential mechanical assembly. With this broader aim in mind, a more concrete aim is to introduce the concept of embryological differentiation into Genetic Algorithms. Traditionally, GAs have contained such ideas as reproductive fitness, crossover, mutation, etc, but nothing equivalent to the way genes switch on and off in natural embryogenesis. The ideas of differentiation are used in this paper to pursue a third aim, namely to try to grow nonconvex artificial shapes of colonies of cells in reproductive cellular automata. Earlier publications aimed at creating an "Artificial Embryology" resulted in the growth of successful convex shapes but nonconvex shapes failed to evolve [e.g. de GARIS 1992].

The new field of Artificial Embryology (or at least the self assembly of complex systems) is felt to be important for the future development of complex system design. New technologies, such as WSI (Wafer Scale Integration), Molecular Electronics, and Nanotechnology promise to allow the construction of devices with a huge number of components (e.g. billions, trillions, and up to Avogadro's number). This in turn will create problems for the design and construction of these devices. If a device contains literally trillions of components, it cannot be built by a machine which positions one component at a time, because such a machine would be too slow. The device will either have to be built by positioning many components simultaneously, (which will become increasingly difficult as the number of components to be simultaneously placed increases), or, the device will have to build itself in a form of embryological self assembly.

It is this second option which is of interest to the authors. It is also the solution employed by nature to build its (hyper) complex systems, such as embryos and brains. However, if one is to employ self assembly techniques to build complex systems, how can one be sure that the self assembled product is useful to human beings? The self assembly will need to be tested for its quality of performance. Those self assemblies which do poorly can be abandoned. Those that do well, may be used. But what if all the initial attempts at self assembly do poorly? How is progress in the design to be made when the self assembled device is massively complex?

The answer to this question may be to mimic nature by using a form of "applied evolution" called Genetic Programming (GP), i.e. using GAs to build/evolve complex systems [de GARIS 1990, 1991a,b, 1992]. Improvement in a hypercomplex system may be achieved blindly by randomly mutating linearly coded instructions for self assembling devices. Those mutations which are "positive", will produce devices with superior performance values. The linear codes which contain these positive mutations can be allowed to produce more offspring in the next generation of a population of such codes.

The authors feel that it is important to begin investigating how to build and evolve (GP style) self assembling devices. This paper is an attempt in this direction. It is part of a general "vision" that future devices may be used to "grow" complex systems, such as electronic circuits at various levels of granularity. At a very basic level, simple RLC circuits may be coded and grown on an electronic substrate in a special hardware device called a "Darwin Machine". At a higher level of granularity, circuits of component parts such as flips-flops and amplifiers could be grown. At a still higher level of granularity, circuits of artificial neurons and synapses could be grown etc. Thus self assembling (and self testing) artificial nervous systems might be grown in these Darwin Machines. This is the vision and the longer term direction in which the work of this paper is aimed.

With this vision in mind, present day computer technology should begin raising its sights beyond the usual style of "homogeneous" architectures. A homogeneous architecture is defined to be one which contains a large number of copies of a small number of relatively simple (i.e. humanly understandable) components or modular designs, which are then connected using simple rules to make the whole. The net result is a machine whose functioning is humanly understandable, because it is built according to a plan, a blueprint. Obvious examples are mass computer memories, von Neumann computers, the Connection Machine [HILLIS 1985], etc. A "heterogeneous" computer architecture is "non homogeneous", e.g. it may contain complex modules which differ greatly one from another, and be connected in very complex ways. The paradigm example of such an architecture is the human brain.

If, for example, one is attempting to build a million or a billion processor machine, (assuming that the underlying technologies allow such a thing), then why restrict oneself to a homogeneous architecture, when one could "almost" as easily (in purely technological implementation terms) build a heterogeneous machine, with all its greater sophistication and subtlety of behavior? Of course, the obvious answer to this question is that a million (billion) processor heterogeneous machine would be hypercomplex and would compare to the complexity level of simple biological nervous systems, in terms of dynamics, structure and most importantly, "un-understandability".

Traditionally, engineers and scientists have shied away from building hypercomplex machines, because of this "un-understandability", i.e. a lack of theoretical principles to explain their structures or functions. However, recently, a new approach to building (hyper) complex systems has been demonstrated. It is called "Genetic Programming" (GP) [de GARIS 1990, 1991a,b, 1992], which uses Genetic Algorithms (GAs) as a tool to build things, where the internal complexity of the system being built/evolved is (within certain limits) irrelevant to its successful construction. So long as the GA being used gets a fitness value which continues to increase over the generations, (i.e. the system has a non zero "evolvability") then a steady improvement in the system being evolved will result. One does not care about the internal complexity. The system is in effect, a "black box". It is possible that this GP approach to building (hyper) complex systems may have an excellent future, and play a vital role in 21st century technology and engineering. It is essentially the approach taken by nature to build/evolve such hypercomplex systems as ourselves.

If the prospect of a one million (billion) processor computer raises the issue of homogeneous vs. heterogeneous architectures and the problem of complexity of heterogeneous machines, then this level of complexity pales into insignificance when one considers the molecular scale technologies now exploding in research labs around the world. Our lab at ETL has whole sections of people devoted to building and testing devices which can pick up atoms at one point and can place them at some other point to build desired molecular scale structures or tools. Molecular scale technologies ("nanotech" [DREXLER 1992]) offer the long term prospect of building Avogadro Machines, i.e. machines containing an Avogadro number (i.e. of the order of a trillion trillion) components. With such a huge number, it will obviously be impossible to assemble them one by one. Somehow they will have to self-assemble, in an embryological like way, as mentioned earlier. Thus the reasoning behind the need for self-assembling, self-testing, GP-style design and construction techniques, will be all the more compelling when it comes to 21st century nanotechnology.

This paper attempts to take some initial steps at using GP techniques to build/grow structures in an embryological like manner, and in particular to use ideas analogous to the phenomenon of "differentiation" in the biological world. It is hoped that the initial ideas presented in this paper, will inspire later papers to be written (by the author(s) and other researchers) on the embryological "growth" of (neuro) electronic circuits, thus creating a new speciality called "Embryonics" (Embryological Electronics). A device which "GPs" (i.e. evolves) a complex (neuro) electronic circuit has been called a Darwin Machine [de GARIS 1990, 1991a] by one of the authors. It is envisioned that such machines may play a role in building artificial nervous systems for future artificial creatures (or biots, i.e. biological robots).

The artificial "embryos" presented in this paper are two dimensional "shapes" formed by a colony of "cells" in reproductive cellular automata. The basic idea is that a GA is used to evolve sets of reproduction rules of these cells. These rule sets are spread out (one set per "operon") along a bit string GA chromosome, and are switched on and off according to whether the state of a cell matches the "condition" field of the operon, and if so, the corresponding "action" field controls the reproduction (or not) of the cell. One has in effect a type of "production system" on a chromosome, which is similar in some ways to Holland's "Classifier System" ideas [HOLLAND 1986].

The remainder of this paper is structured as follows. Section 2 provides a more general introduction to differentiable chromosomes and shape genes. Section 3 contains a more detailed description of the algorithms used in this paper to grow nonconvex shapes. Section 4 presents some results of experiments using the ideas of the previous section.

2. Differentiable Chromosomes and Shape Genes

FIG. 1 shows what the authors mean by a "differentiable" chromosome, which in this particular example, consists of four genes (or "operons", a term taken from molecular biology), where each operon (separated by double vertical lines) contains a condition field C and an action field A . These operons switch on and off over time, as in nature. The actions of earlier operons cause other operons to switch on and off. The net result of this sequential operon switching can be the controlled growth of an (artificial) embryo.

Each cell of a particular cellular automaton described in this paper contains the same chromosome. At each stroke of a clock (where the cellular automata are assumed to be synchronous or clocked), each cell calculates its present "state" by observing the states of its neighboring cells at the end of the previous cycle (where a cycle is the time between two clock strokes). Each cell then compares its present state with all the coded condition fields C_i of its genes (or operons). If one of the conditions (e.g. C_m) "matches" sufficiently, then the corresponding action field A_m is "activated" or "expressed", and the instructions coded in that action field are executed. The activation of these instructions may change the state of some of the cells, and thus activate new instructions, which in turn may change further states etc. Thus it is possible to generate a time sequence of state

changes and execution of instructions, which result in some final desired product, e.g. an embryo or colony of cells having a desired shape.

The ideas contained in the previous paragraph are very general, and may be applicable to many kinds of concrete situations. The above ideas have similarities with traditional "production systems" of Artificial Intelligence and Holland's Classifier Systems [HOLLAND 1986]. Perhaps future research might be able to incorporate ideas from the extensive work already done on Production and Classifier Systems into differentiable chromosomes.

In order to give a more concrete example of what coded instructions might be like, a brief introduction to some earlier work of the first author on Artificial Embryology will now be given [de GARIS 1992]. Basically, the main idea of this earlier work was to evolve reproduction rules for cellular automata (CA), such that the final shape of a colony of cells attained, as closely as possible, some desired shape, such as a square, triangle, ellipsoid, tadpole, etc. The state of a cell in one of these experiments was defined in terms of the configuration of its neighborless sides. Assuming only edge cells can reproduce, (where reproduction is defined as putting a daughter cell in a vacant square contiguous with the parent cell), there are 14 possible states, as shown in FIG. 2.

For a 2D cellular automaton, each cell has one of the above 14 states (assuming only edge cells can reproduce, and that no isolated cells are generated). For each cell, for each cycle, a cell can reproduce or not, hence 14 bits (one per state) are needed to specify which cells in a given cycle will reproduce, as shown in FIG. 3 with the REPRO? fields (one per cycle). If a cell is to reproduce, it needs to know in which direction (E, N, W, or S). If there is only one neighborless side, then no REPRO? field bits are needed to specify the side for the daughter cell, because there can only be one choice.

For two neighborless sides, 1 bit is needed (for 6 such states). For 3 neighborless sides, 2 bits are needed (for 4 such states). If an occupied side is specified with these 2 bits, it is ignored. Hence to specify the reproduction directions, ${}^4C_2*1 + {}^4C_3*2 = 14$ bits are needed, as shown in FIG. 3 by the R DIRNSi fields. Thus, $14 + 14 = 28$ bits per cycle are used, with separate REPRO? and R DIRNS fields for each cycle. The number of cycles (or iterations) NI is also coded onto the bitstring chromosome and evolves along with the other fields.

These chromosomes are then evolved using a Genetic Algorithm [GOLDBERG 1989], such that the resulting colony of cells attains as closely as possible some desired or target shape. The fitness of the chromosome (i.e. the measure of closeness of the final and the target shapes) was defined as follows :-

$$\text{Fitness} = (\#ins - 0.5*\#outs)/(\#des)$$

#ins = the number of **filled** cells inside the desired shape
#outs = the number of **filled** cells outside the desired shape
#des = the number of cells inside the desired shape

As an example of this type of evolution (for a triangular target shape), see FIG. 4, which shows the result at each of 4 cycles, i.e. NI evolved to be 4.

The above ideas were tested on various shapes [de GARIS 1992], both convex and non-convex. The convex shapes worked well (with fitness values around 95%, e.g. FIG. 4), but non-convex shapes evolved poorly, with low fitness values (e.g. as low as 20% for an arch shape). The challenge is to evolve arbitrary non-convex shapes, e.g. 3D embryo shapes with a head, body, limbs, fingers and toes etc. Evolving such an "artificial embryo" implies a type of sequential, time dependent "unfolding" of shapes, e.g. first the body is grown, then the limbs and head emerge from the head, then the fingers and toes emerge from the limbs etc. It appears that one needs some means of switching on and off shape forming instructions for the various components.

3. Evolving a (Non-Convex) "L" Shape

To illustrate the ideas contained in the above section, a concrete example is now introduced, which is to "grow" a simple nonconvex target shape in the form of the letter

"L". FIG. 5 shows the set up. To keep things simple, it is assumed that there are only 2 operons in the "L chromosome", and that the first operon is used to code for instructions to grow region A, and the second operon is used to code for region B. The trick then is to decide how to define the state of a cell, plus the nature of the condition fields, so that operon "A" switches off at the appropriate time and operon "B" switches on. What is required is that once the region A has been filled, all the edge cells (except those in region R) stop reproducing. The R cells then grow region B. Hence we need some means to distinguish the R region from the rest of the edge cells of region A (assuming only edge cells reproduce). The R region lies south-east of the A region, so one idea might be to let the A region grow for a given number of iterations (as in the above section) and then let only the south-east cells reproduce beyond that number. So, the condition for region A might be simply :-

$$[\text{nbr of iterations} = < X]$$

where X needs to be evolved. The condition(s) for region B might be :-

$$[\text{nbr of iterations} = X] \& [\text{cell region type} = \text{south-east}]$$

OR

$$[X < \text{nbr of iterations} = < X+Y]$$

Alternatively, the region B operon could be split into 2 operons B1 and B2, where the condition for B1 would be the first disjunct, and the condition for B2 would be the second disjunct. What are needed now are means to specify the number of iterations, and "south-east-ness". (Note that in an iteration, all edge cells in a colony of cells which are allowed to reproduce, do so). One idea to specify something similar to the number of iterations (in cellular terms), might be to give each cell a "generation count" (GC), such that a parent cell with a GC of "g", would have a daughter cell with a GC of "g+1".

To specify "south-east-ness", one might use a concentration gradient of some "chemical", which is passed in (fractionally) decreasing quantities from parent to daughter cell, e.g. a parent cell may have a quantity "q" of this chemical, which is fractionally reproduced and transmitted to the daughter cell which receives (for example) a quantity "0.95q" of this chemical. Assume that each cell uses this chemical to create another chemical which is diffused into the intercellular environment. By measuring the "q value" of neighboring cells, each cell can determine the "q gradient", and hence determine its "position".

To calculate the "q gradient" at each cell, the following approach is suggested, as shown in FIG. 6. A square block of 49 cells, i.e. the 7 by 7 "neighborhood" of a cell is used. The average "q value" over 4 cells is calculated (i.e. over the "middle" cell, and the other 3 cells in the same direction). For example, if the middle cell is given coordinates (i,j) then the average "q value" in the north-west direction (Q_{nw}) would be :-

$$Q_{nw} = 0.25 * [q(i,j) + q(i-1,j-1) + q(i-2,j-2) + q(i-3,j-3)]$$

Similar Q values are calculated for all 8 directions (i.e. E(ast), NE, N, NW, W, SW, S, SE). Since these calculations are only performed at the edge cells, about half of these 8 directions will involve empty cell positions. Hence the average "q value" in such directions will be low. The high "q value" directions are likely to be towards the "center" of a colony of cells, i.e. perpendicular to the edge of the colony at the point (i,j). The "q gradient" (QG) at a cell, is defined to be the direction (one of the 8 possible) which has the largest q value. For example, those cells lying at the south-east edge of a colony of cells, will have their "q gradients" pointing north-west.

The notions of "generation count" and "q gradient" can be used to decide when cells should switch from their first operon to their second. What now needs to be discussed is how these notions can be coded on the operon, (i.e. how to express the condition part of the operon) and how the cells are to reproduce, (i.e. how to express the action part of the operon).

The following chromosome format is suggested, as shown in FIG. 7.

This format requires some explanation. X is the generation count (GC) of a cell when its Operon 1 switches off and Operon 2 switches on. A cell with a GC of X+Y, switches off its Operon 2.

The NEWS DIRNS field is 8 bits long, and is used to specify which cells can reproduce beyond X generations. Only those cells whose "q gradients" have their corresponding bits set in this field can reproduce further. For example, if a cell's "q

gradient" is NE, and the bit corresponding to NE is set (i.e. the second bit), and its GC is X, then that cell can use Operon 2 when its GC obeys $X < GC < X+Y$.

REPRO?/DIRN PAIRS (or RD Pairs) are the pairs of fields shown in FIG. 3. The "i"th pair of fields applies to a cell whose GC is "i". The "j"th pair in Operon 2 applies to a cell whose GC is "X+j".

If "m" bits are reserved for the X field, then 2^m RD pairs are needed in Operon 1. Similarly, if "n" bits are reserved for the Y field, 2^n RD pairs are needed in Operon 2. Hence the length of the chromosome can be calculated as $(m + 28*2^m + n + 8 + 28*2^n)$ bits. Note that cells are not obliged to switch from Operon 1 to Operon 2 simultaneously, but only when each cell reaches a GC of X. The fitness definition is the same as defined earlier, i.e. $\text{Fitness} = (\#ins - 0.5*\#outs)/(\#des)$, where the target shape is the "L" (double rectangle) of FIG. 5.

4. Experimental Results

This section presents some experimental results of the ideas introduced in earlier sections. However, some changes were made for programming convenience. For example, the evolved values X and Y were no longer interpreted as the generation counts (GC) of individual cells, but simply the number of reproductive iterations or loops in the algorithm used in the computer program. Thus X becomes the number of loops using operon 1, and Y becomes the number of loops using operon 2.

These results were obtained from programs run on a MAC Iici computer, in "C" language. Two general approaches to the evolution were usually taken. In the first approach, both operons were switchable during a "run" (i.e. the second operon was able to switch on later in the run). In the second approach, so called "shaping" techniques were used. The concept of "shaping" has been defined in earlier publications [de GARIS 1991a,b, 1992]. "Shaping" is simply splitting up an evolutionary process into intermediate phases, with intermediate targets. The results of a previous phase (i.e. an evolved chromosome), are fed as starting conditions (i.e. initial chromosomes) for the next phase. This is done to reduce the size of the search space, and thus accelerate the evolution. In some cases, shaping is needed just to get a required evolution.

In a first experiment, no shaping was used, i.e. both operons were allowed to be active, starting with random chromosomes. FIG. 8 shows the results after 600 generations, and little further fitness growth. The X and Y values evolved to be 13 and 15 respectively, the NEWS DIRNS were [00100010]. The fitness was about 80%. FIG. 8a shows the colony of cells grown using the instructions of the first operon (i.e. before the switching to the second operon). FIG. 8b shows the final result after both operons have finished, i.e. after X+Y iterations.

An 80% result was not thought to be particularly impressive, although at least a definite "L" shape began to emerge. A second experiment was undertaken using shaping techniques to see if a better result could be obtained. In this second experiment, initially only operon 1 was allowed to evolve (using the L target shape). Operon 2 was switched permanently off. In a second phase of evolution, the resulting chromosome population from the evolution of operon 1 was used as a starting set of chromosomes, in an evolutionary phase in which operon 2 was able to switch on.

FIG. 9 shows the results using shaping techniques. FIG 9a resulted from 1200 generations of evolution of Operon 1 alone. FIG 9b resulted from taking operon 1 from FIG. 9a and allowing it to evolve a further 1500 generations with both operons. FIG. 9b shows that a definite turning occurred before fitness values stagnated, and that a nonconvex shape is being formed.

For FIG. 9a, the X value evolved to be 15, and in FIG. 9b, both the X and Y values evolved to be 15. For FIG. 9a, the NEWS DIRNS were [11000110], which evolved to [01000100] for FIG. 9B.

The final fitness value of FIG. 9b was also about 80%, so shaping did not seem to have much of an effect. The vertical part of FIG. 9a is deeper and "cleaner" than in

FIG. 8a, as one would expect, but the net result in FIG. 9b is not particularly praiseworthy.

These 80% results were rather disappointing, and showed the difficulty of the task. With convex shapes (circles, squares, rectangles, ellipses) in earlier experiments [de GARIS 1992] fitness values around 95% (using the same fitness definition as in this paper) were typically obtained, but nonconvex shapes failed badly. The results of FIGs 8 and 9 show that at least nonconvex shapes can be obtained, but not (yet?) to high quality. This relative failure is probably not surprising with hindsight. The algorithm as presented in this paper, effectively causes one convex shape to grow out of a subset of edge cells of a previous convex shape. By having a series of switch-ons and offs, one could probably generate some interesting shapes, but fully random shapes will probably require new kinds of algorithms.

A third experiment using the "L" target shape was undertaken, which gave better results. The four seeder cells were moved down 6 squares, so that they were positioned more centrally relative to the vertical part of the L. This seemed to constrain the evolutionary path less. The X and Y values which evolved in FIG. 10 were 10 and 15 respectively, and the NEWS DIRNS were [01100011]. Note that FIGs 10a and 10b are from the same run, with no shaping. The resulting fitness value was 87%.

The parameter values used for the above experiments were as follows :-

Population Size	20 chromosomes
Chromosome Length	1810 bits
MaxX	32
MaxY	32
Max Number of Bits for X Field	5
Max Number of Bits for Y Field	5
Seeder Cells Chemical Q Value	1.0
Chemical Transmission Coefficient	0.95
Cell Grid Size	48*48 cells
Seeder Cell Coords (FIGs 8, 9)	(20,14), (21,14), (20,15), (21,15)
Seeder Cell Coords (FIG 10)	(20,20), (21,20), (20,21), (21,21)
(Uniform) Crossover Probability	0.6
Mutation Probability	0.005/bit
Linear Scaling Constant	2.0

To conclude this paper, the authors would like to illustrate further, just how difficult it is to grow more complex and interesting nonconvex shapes. FIG. 11 shows the target shape of a turtle. It was thought that the first operon might grow the circular body, and that the N, NW, NE, SW and S edges of the body (corresponding to NEWS DIRNS of [01010111] (see FIG. 12)) would switch on, and grow the head and 4 limbs with the second operon.

The best actual result obtained after 200 generations and little further fitness improvement is shown in FIG. 13 (where X and Y evolved to be 17 and 27 respectively, and the NEWS DIRNS was [11001101]. The fitness was only 80% and does not look anything like a turtle. It looks more like a blob. This is probably because the X value is too small and the Y value too large. If the switching had occurred later, the "Y shapes" might have been more limb like.

Perhaps a better result could be obtained by using shaping techniques. For example, a circular target shape could be used with only operon 1 allowed to be active. Previous experience shows that generating good circular shapes is not difficult. In a second phase of evolution (with the full turtle target shape), and using the chromosome evolved from the first phase, hopefully a rather small Y value and the desired NEWS DIRNS [01010111] would be evolved to generate the turtle.

In all of these experiments, there has been only one switching, and many iterations. Perhaps one could increase the number of switchings and thus obtain better shapes. In any event, it is hoped that the challenge to "grow" artificial embryos will appeal to other ALife researchers, so that some real progress can be made in this challenging field in the next few years. After all, as was expressed rather strongly in the introduction, future molecular based technologies will have to become self assembling, i.e. such a step is felt by the authors to be inevitable, so some form of "Artificial Embryology" becomes essential.

The authors next research project along these line will be to attempt to "grow" electronic circuits, using GP techniques, i.e. to establish a speciality called "Embryonics" (Embryological Electronics).

References

- [de GARIS 1990] de Garis H. (1990), "Genetic Programming : Building Artificial Nervous Systems Using Genetically Programmed Neural Network Modules", in Porter B.W. & Mooney R.J. eds., Proc. 7th. Int. Conf. on Machine Learning, pp 132-139, Morgan Kaufmann.
- [de GARIS 1991a] "Genetic Programming : Artificial Nervous Systems, Artificial Embryos and Embryological Electronics", Hugo de Garis, in "Parallel Problem Solving from Nature", Lecture Notes in Computer Science 496, Springer Verlag, 1991.
- [de GARIS 1991b] "Genetic Programming", Hugo de Garis, Chapter 8, in book, "Neural and Intelligent Systems Integration", ed. Prof. Branko Soucek, WILEY, 1991.
- [de GARIS 1992] "Artificial Embryology : The Genetic Programming of an Artificial Embryo", Hugo de Garis, Ch. 14 in book "Dynamic, Genetic, and Chaotic Programming", ed. Branko Soucek and the IRIS Group, WILEY, 1992.
- [DREXLER 1992] "Nanosystems : Molecular Machinery, Manufacturing and Computation", Drexler K.E., Wiley, 1992.
- [GOLDBERG 1989] "Genetic Algorithms in Search, Optimization, and Machine Learning", Goldberg D.E., Addison-Wesley, 1989.
- [HILLIS 1985] "The Connection Machine", Hillis W.D., MIT Press, Cambridge, Mass, 1985.
- [HOLLAND 1986] "Escaping Brittleness : The Possibilities of General Purpose Learning Algorithms Applied to Parallel Rule-Bases Systems", chapter 20 of, "Machine Learning: An Artificial Intelligence Approach, Volume 2", R.S. Michalski, J.G. Carbonell, T.M. Mitchell eds., Morgan Kauffman, 1986.

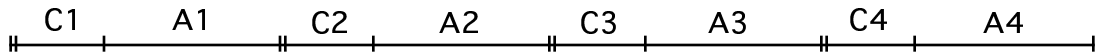


FIG. 1 A DIFFERENTIABLE CHROMOSOME

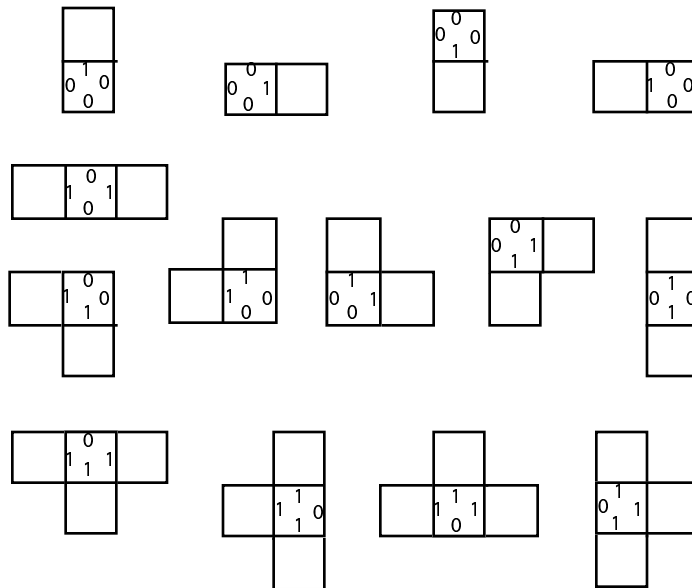
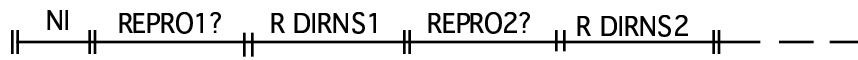


FIG. 2 The 14 Neighborless Patterns for 2D Cells



NI = NUMBER OF ITERATIONS

REPRO1? = WHICH STATES CAN REPRODUCE IN ITERATION 1

R DIRNS1 = REPRODUCTION DIRECTIONS IN ITERATION 1

REPRO2? = WHICH STATES CAN REPRODUCE IN ITERATION 2

R DIRNS2 = REPRODUCTION DIRECTIONS IN ITERATION 2

FIG.3 FORMAT OF A SIMPLIFIED "EMBRYO" CHROMOSOME

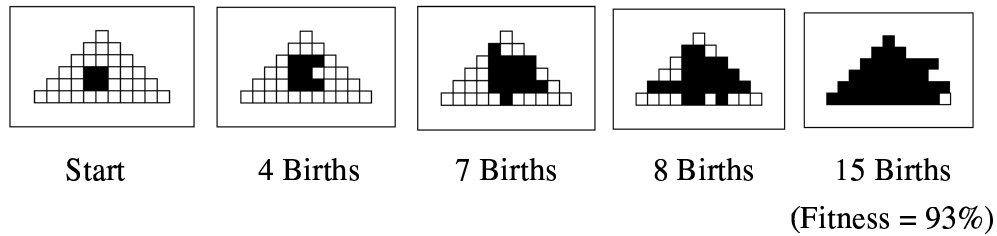


FIG. 4 TRIANGULAR TARGET SHAPE

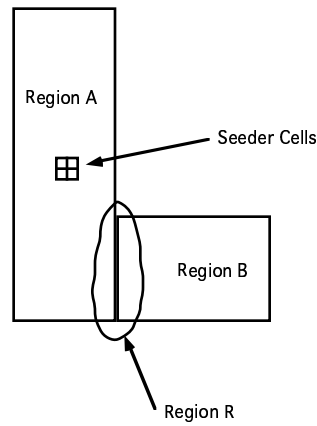


FIG. 5 A (NON CONVEX) TARGET SHAPE "L"

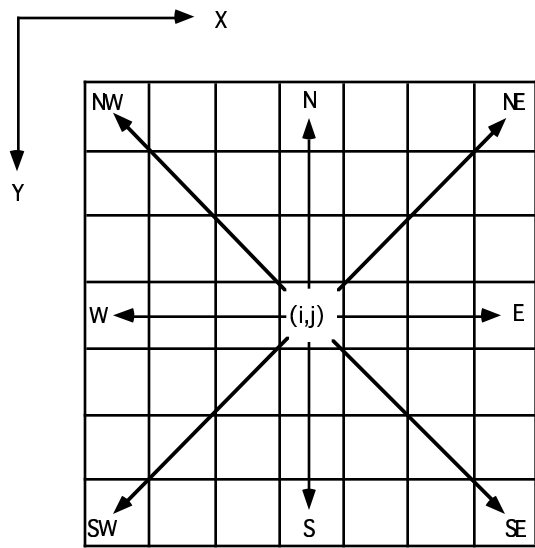


FIG. 6 CELL NEIGHBORHOOD

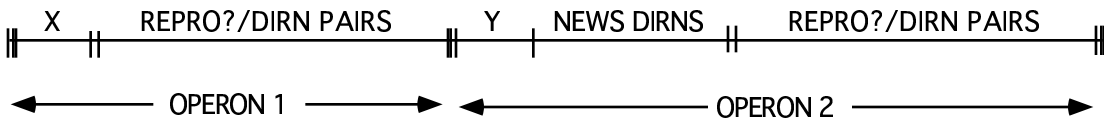
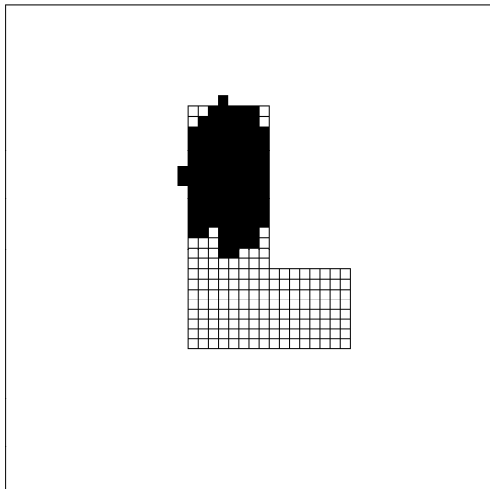
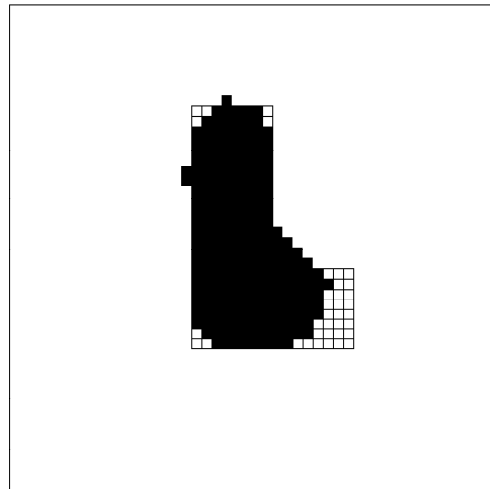


FIG. 7 CHROMOSOME FORMAT

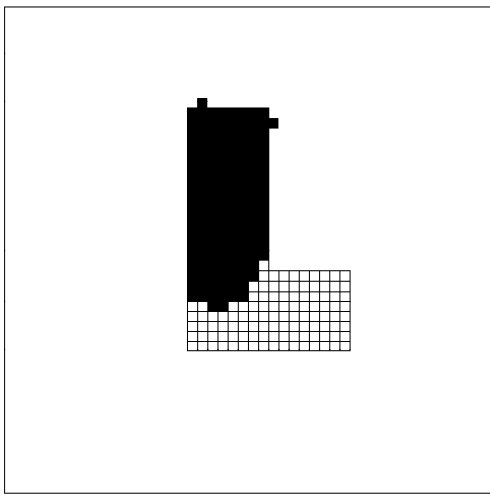


(a) AFTER OPERON 1

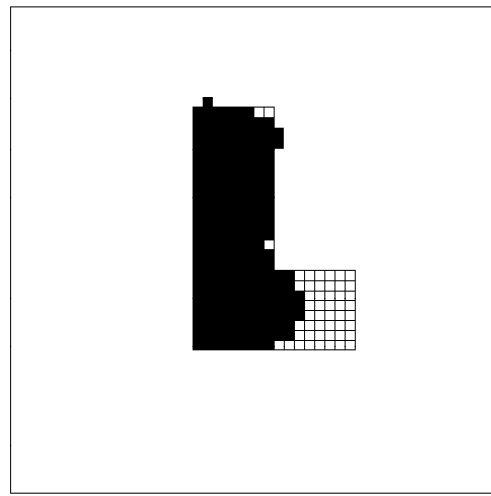


(b) AFTER OPERONS 1 & 2

FIG. 8 "L" SHAPE RESULTS USING BOTH OPERONS

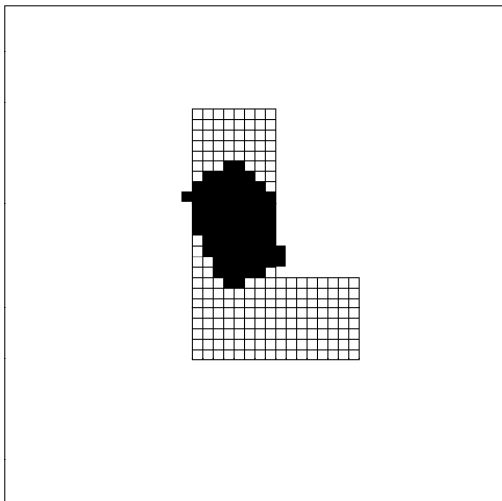


(a) OPERON 1 ONLY

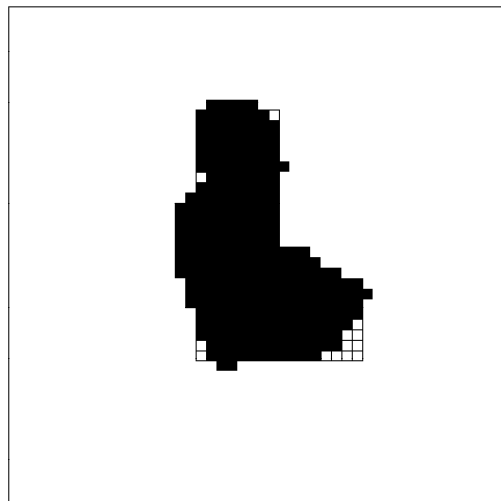


(b) AFTER OPERONS 1 & 2

FIG. 9 "L" SHAPE RESULTS USING SHAPING TECHNIQUES



(a) AFTER OPERON 1



(b) AFTER OPERONS 1 & 2

FIG. 10 "L" SHAPE RESULTS WITH MORE CENTRAL SEEDER CELLS

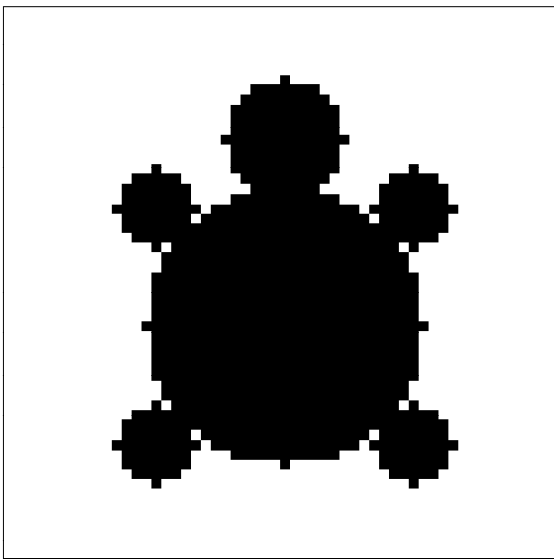


FIG. 11 TURTLE TARGET SHAPE

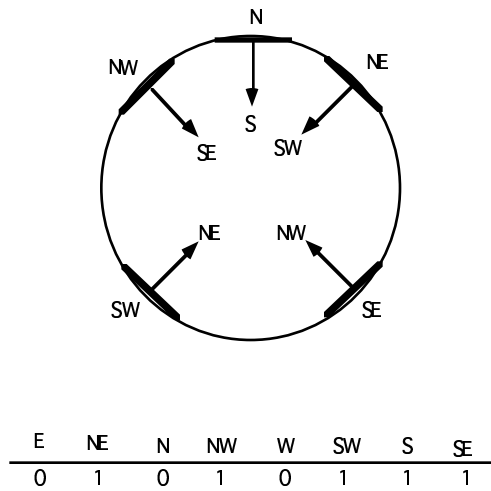
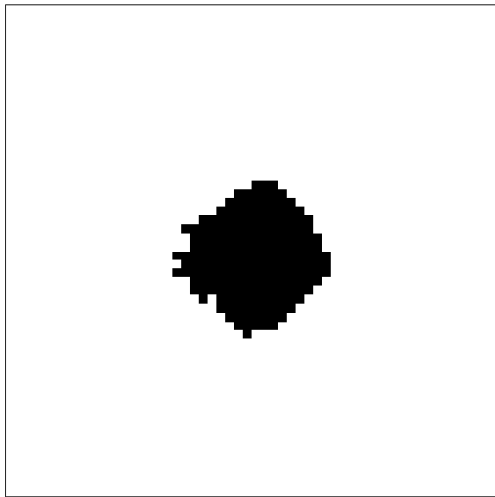
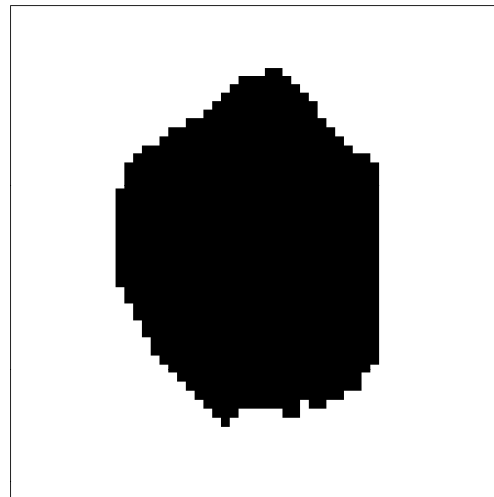


FIG. 12 DESIRED NEWS DIRNS



(a) AFTER OPERON 1



(b) AFTER OPERONS 1 & 2

FIG. 13 ACTUAL TURTLE SHAPES