Size Matters: Scaling of Organisms and Genomes for Development of Emergent Structures

Gunnar Tufte
Norwegian University of Science and Technology
Department of Computer and Information Science
Sem Selandsvei 7-9 7491 Trondheim Norway
gunnart@idi.ntnu.no

Joacim Thomassen
Norwegian University of Science and Technology
Department of Computer and Information Science
Sem Selandsvei 7-9 7491 Trondheim Norway
joacimt@stud.ntnu.no

ABSTRACT

An artificial Development approach aimed at development of electronic circuits has functional circuits as the end product. Functionality of circuits are given by the topology of the phenotype. The article investigates if the iterative processing of the genome in a development process can take advantage of the information provided in the genome and the information available in the emerging phenotype. The development process described is a rule-based system on a non-uniform Cellular Automata topology. The experiments presented investigate scaling of available resources in the phenotype for evolution of structural properties and genome scaling for expressing functionality in the structure of the emerging phenotype.

Categories and Subject Descriptors

B.6.1 [**Hardware**]: Design Styles—*Cellular arrays and automata*

General Terms

Design

Keywords

Development, Cellular Computation, Scaling

1. INTRODUCTION

In Evolvable Hardware (EHW), Evolutionary Algorithms (EAs) are used to evolve electronic circuits. In general, a one-to-one mapping for the genotype-phenotype transition is assumed. Use of a one-to-one mapping assumes a genotype consisting of a complete blueprint of the phenotype. For an electronic circuit, the genotype thus completely describes the circuit topology i.e. the components and their interconnections. This description may be said to be a structural description. Interpretation of this structural information provides us with the circuit functionality. Electronic circuits may be said to be complex due to their complex structure and/or complex functionality.

The approach of evolutionary design [3] may be an alternative to today's view on how circuits and computation machines are designed and operates. If an alternative design approach is to be used it may be that today's view on a machine's components and architecture constrains the design approach [16].

The Cellular Computation paradigm [21] may be an alternative architecture. Cellular Computation offer a theoretical platform [20] realisable in today's hardware technology [27] exploiting the principles of a global behaviour emerging from local cell interactions in a parallel architecture.

The massive parallel computation power of cellular computation is hard to exploit using traditional design and programming methods. Moving away from a traditional approach towards adaptive methods [21], includes systems where the programmer (or designer) can not explicitly specify the complete system [20]. An adaptive approach, e.g. EAs, may be suited but constraints of the algorithm itself can be added into the design process. EAs are resource-greedy, accompanied by direct mapping and often suffer from a scaling problem [4].

One solution to the resource greedy nature of EAs is to follow nature's example and shrink the genotype in some way. Natures way of handling complexity clearly points in the direction of a non one-to-one mapping from genotype to phenotype. Natures process of development where a zygote develop to a multicellular organism [30] can be included in an EA to increase scalability [2].

In biological development, an initial unit — a cell, holds the complete building plan (DNA). It is important to note that this plan is generative — it describes how to build the system, not what the system will look like. Units have internal state, can communicate locally, can move, spawn other units or die. Groups of units may also exhibit group-wise behaviour i.e. a group state.

The global developmental stages from the zygote (fertilised egg) to the multicellular organism, although interdependent and not strictly sequential, may be categorised as *pattern formation; morphogenesis; cell differentiation and growth* [30].

Evaluation of individuals in a developmental system is usually based on properties of the phenotype i.e. the final developed phenotype or the emerging phenotype. In contrast to a one-to-one mapping where the phenotype may be identical to the genotype [12]. The genotype in a development system may only provide information regarding the phenotype or it's functionality as the genome is processed by the development process.

For development of structures e.g. flags [17], a graphical representation of a flag is the output produced. Evaluation is based on structural properties of the phenotype given by cells representing structural parts of a global property e.g. a desired pattern of cells expressing colours. The development process in these flag examples have built a structure out of cell interactions i.e. the global structure emerges out of local interactions.

Development of functionality i.e. a phenotype structure capable of computation, can use an evaluation based on the behaviour of the emerging phenotype. The Cellular Computation paradigm [21] used herein offer massive parallel computation power in a cellular array. The emergent phenotype consists of computation elements in a cellular array. As such, the evaluation can be based on the functional properties expressed in the cellular array i.e. cellular computation [23].

As stated above the processing of genomes in a development system makes it quite different from a system using a one-to-one mapping. The processing of the genome in the development process in the cellular development approach used herein is based on gene regulation [24]. Gene regulation implies that different parts of the genome are expressed in different cells at different time in the emerging phenotype. The phenotype emerges as a result of an interplay between the genome and the emerging phenotype i.e. a process of gene regulation.

Way back in the work of Von Neumann [29] it was shown that the number of available cell types and the number of available construction instructions influence on what cellular structure that can be built. In the work of Sipper [20] the number of available cell types required for general computation is investigated for the purpose of cellular machines. Inspired by such approaches we want to investigate how scaling of available information influence on evolution of developmental genomes.

Available information is herein scaled by two parameters. First, the available size of the phenotype i.e. the number of cells that can be exploited by the growth and differentiation process. Second, the available number of genes in the gene regulation network. The scaling of genome i.e. number of genes, changes the available actions a cell can express and the number of regulation criteria the cell can interpret.

To explore the influence of these two scaling parameters two very different approaches are presented. However both uses the same cellular development process. Two different EAs are used. An Evolutionary Strategy (ES) is used in experimental investigation of the scaling of available cells i.e. phenotype scaling. A Genetic Algorithm (GA) is used to investigate the influence of available genes i.e. genome scaling.

The two experiments presented targets evolution of genomes for structural properties and functional properties respective. The goal of presenting two so different approaches is to show that they have interesting concurrent results that appear independent of the EA included in the system.

The developmental model used in both experiments is based on cellular development. The model include a cell with functional components in addition to the required genome and development mechanisms. As stated the main long term goal is functional circuits. As such, functional cell components are required to express functionality. Expressing functionality adds an extra process to the developmental system. In addition to the processing of the developmental genome to generate the emerging organism the functional components of the emerging organism must be processed.

The resources required to process the developmental genome as to generate the emerging phenotype usually depend on the size of the genome and the amount of development steps. The processing of the genome may be a sequential or parallel process. To express cells' functionality the cell's functional components must also be processed. As such, the amount of computational power required for each cell depends on the processing of the genome and the processing necessary to express the cell's functionality. The total amount of computational power is given by the size of the cellular array, i.e. the computational power required by the total number of cells.

Herein the functional components of the cell are only processed in the experiment of functional properties. The functionality is evaluated on the final development step. However, the functionality expressed in the organism at the final development step arises as a result of processing of the functional components on all available development steps.

To be able to carry out experiments including processing of the functional components and the desired genome size in realistic time a hardware experimental platform has been implemented. The hardware platform offer true parallel computation for expressing the functionality of the cells, i.e. the processing time is not influenced by the amount of cells used. The processing of the genome is partly processed in parallel offering a speed-up making it possible to scale up the genome size and still be able to perform experiments in realistic time for the desired amount of cells.

The speed-up offered by the hardware platform is necessary for the experiments regarding functional properties. In the experiments regarding structural properties the genome size is much smaller and the functional components of the cells are not processed. As such, a software solution is possible. The reason for using both software and a hardware platform in experiments is to be able to exploit the raw power of hardware when necessary and to be able to take advantage of the flexibility offered by software. Flexibility here relates to the ability to do flexible monitoring of internal processes of the development process. However, running a software solution is constrained by increased computation time for large genomes, simulation of functional components and the amount of possible cells.

The article is laid out as follows: The cellular developmental model is presented in Section 2. Section 3 describes the ES used and the experiments for phenotype scaling. The GA used and experimental results for genome scaling are presented in Section 4. A discussion of the experiments are given in section 5. Finally, Section 6 concludes the work.

2. DEVELOPMENT MODEL

Including properties of biological development as described above into artificial development do not need to achieve a realistic model of development but are rather used to increase the power of evolutionary computation or Evolvable hardware (EHW) [6, 14, 18, 26, 28]. However, artificial development may also be used in studies by biologists as demonstrated by Kumar's [10] computational models of development based on, e.g. real biochemical pathways. The complex 3D shape and form in Kumar's experiments, e.g. [11], illustrate evolution together with development which achieve complex structures.

Today's electronic circuits are based on 2D silicon technology. As such, it is possible and desirable to limit the complexity of the shape and form of the target for development. In [6, 26] a 2D circuit close to the internal architecture of FPGAs are used as the target for development of electronic circuits. Both approaches uses a intracellular communication restricted to a 2D Von Neumann neighbourhood. However, the architecture and computation paradigme used is quite different. In Gordons work [8] successful evolution of adders and parity problems was demonstrated. The architecture of the circuit and computation performed may be close to a traditional circuit even if the circuit solutions found are not traditional adder designs. The architecture of the developed circuit consists of combinatorial function generators connected by wiring. As such, the circuit is a combinatorial circuit consisting of inputs, computational elements and outputs. The work herein is based on the development model in [26] using a different computational paradigm, i.e. cellular computation. The computational elements, i.e. functional components of the cell, are a sequential circuit including memory and combinatorial logic. The architecture of the developmental circuits is based on the cellular computation paradigm. The circuits functionality is an emergent property. As such, the work of Gordon targets development of combinatorial circuits the work herein is aimed towards development of emergent sequential circuits.

In [7] Gordon applies the developmental model to the task of evolving pattern to demonstrate scalability. The results show that the development model for electronic circuits also can be applied to develop patterns. Herein a similar approach is taken. The development model is applied to the task of development of a specific goal pattern. However, the possible cell types are restricted to include only the required types i.e. the number of exploitable cell states (given by the cell types) are at a minimum. In contrast the possible cell types (or cell states given by the possible protein concentration) in [7] are not constrained as the goal is scaling of the phenotype itself. Scaling herein investigate scaling of available resources in the phenotype for evolution of structural properties.

Figure 1 shows the building block of the developmental system presented herein — the cell. The cell is divided into three parts: the genome (the building plan); the development process (mechanisms for cell growth and differentiation) and the functional component of the cell. The genome consists of rules of how to construct the multicellular organism i.e. a cellular array of functional components.

The genome is based on two types of rules i.e. change and growth rules. Cell growth is a mechanism to expand the organism and differentiation changes a cell's type i.e. functionality. These rules are restricted to expressions consisting of the type of the target cell and the types of the cells in its Von Neumann neighbourhood. The rules consist of a result and a condition. The result part of a change rule states the type of cell the target is going to be changed into. The conditional part describes the type of neighbourhood cells to trigger the rule. A growth rule result gives the direction of growth; grow from north **Gn**, east **Ge**, south **Gs** or west **Gw**. Rules have the following valid conditions: valid cell types, don't care (DC), or empty. DC is not a valid target condition.

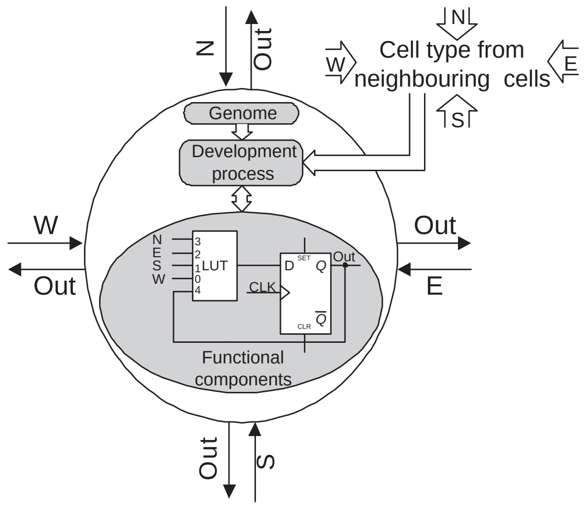


Figure 1: Components of the cellular Development Model.

Change rules have one restriction: a target cell can not be changed from an empty to a valid non-empty cell type. The reason behind this restriction is that we want growth to handle the expansion of the organism. Growth rules have two restrictions. First, the target cell must be empty – this is to prevent growing over an existing cell and, therefore, specialising the cell with a new cell type. Secondly, the cell to be copied into the target can not be empty.

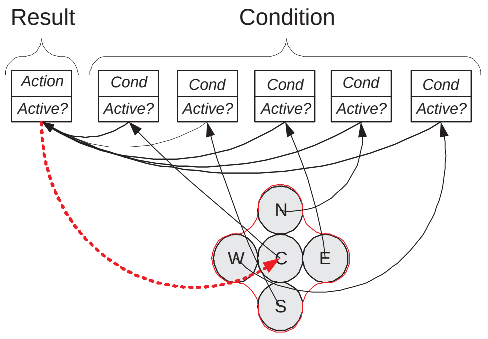


Figure 2: Regulation of genes as interplay between the genome and the emerging phenotype.

Firing of a rule can cause the target cell to change type, die (implemented as a change of type) or cause another cell to grow into it. The current cell together with the neighbouring cells control whether a rule is to be fired or not. Figure 2 illustrates the process of evaluating a rule. For each cell condition, the cell type is compared and if the condition is true then that part of the rule is active. If all conditions are active then the result will become active and the rule will fire. Activation of the result gene is expressed in the emerging phenotype according to the action specified in the result.

In a development genome multiple rules are present. Multiple rules imply that more than one rule of a given cell may be activated at the same time if their conditions hold. To ensure unambiguous rule firing, rule regulation is part of the development process. If the first rule is activated, the second and third rule can not be activated. Activation of the second rule prevent activation of the third rule, etc i.e a gene regulation network.

The development process presented is an autonomous process. All cells in the cellular array can run the development process in parallel. The functional components of the cells are also a parallel architecture. As such parallel cellular development starting from a single cell can develop to a multicellular functional organism i.e. a cellular array constructed of different cell types.

The functional components of the cell is an Sblock [9]. The context of the Sblock's look-up table (LUT) defines it's functionality and is herein also used to define Sblocks as cell types based on their functionality. The LUT is the combinatorial component and the flip-flop is the memory element capable of storing the cell state. The output value of an Sblock is synchronously updated and sent to all its four neighbours — its Von Neumann neighbourhood, and as a feedback to itself.

One update of the cell's type given by the genome and the execution of the development process is termed a development step (DS). A development step is a synchronous update of all cells in the cellular array. The update of the cell's functional components i.e. one clock pulse on the flip-flop, is termed a state step (SS).

The POE-model [19] has been established as a taxonomy of biological inspired hardware. The POE-model classifies biological inspiration in the design of computing machines along three axes: phylogeny, ontogeny, and epigenesis. The phylogeny axis encompasses evolution. Ontogeny embraces systems taking inspiration from biological development. Systems capable of acquiring and exploiting information i.e. learning, are placed along the epigenesis axis. A common property for designs exploiting ontogeny and epigenesis, is the ability to shrink the level of information needed to form a large complex organism [28].

A possible approach to hardware development of cell based circuits is to include the genome, functional units and development process in each cell [14, 15]. In the POEtic project the goal is to include all of the three axes of the POE-model in digital hardware. Others have included the evolutionary process in the hardware platform [13] targeting fault-tolerance.

Another approach is to simplify each cell, in keeping with the properties of cellular architectures, by removing the genome and development process from the cell itself [25] thus increasing the parallelism available to the functional parts of the cell. The principle of a common genome is still retained as all the cells' development actions are controlled by the same shared genome even if it is not stored in every cell.

Figure 3 illustrates the hardware implementation of the cells with a centralised genome and development process used in section 4. The fixed partition consists of a communication module (*COM*) for external communication and genome download. The *CTRL module* manages the other modules. The different Cell types, i.e. LUT definitions are stored in the *Sblock definition* module. The genome is stored in the *Rule Memory*. The *Development Process* updates the emerging phenotype based on the properties of the emerging phenotype and the genome. *CONFIG* is the interface

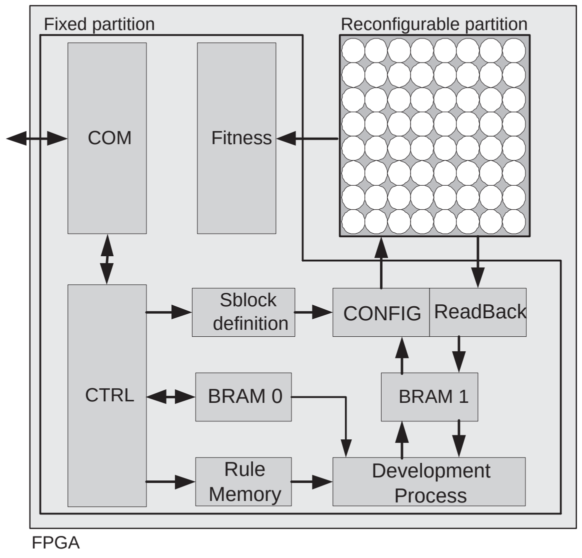


Figure 3: Development process integrated on-chip together with an Sblock array.

to update the phenotype in the reconfigurable partition of the FPGA. The *ReadBack* module can be used to collect information from the phenotype. The emerging phenotype develops in the reconfigurable partition. Each circle include the functional components of the cell shown in Figure 1. The *Fitness* module can be used to implement hardware fitness functions [1]. A detailed description of the modules can be found in [25].

3. PHENOTYPE SCALING

The objective is to evolve genomes that can produce an emerging phenotype using structural properties of the phenotype as the main evaluation criteria. The structural property chosen is development of a predefined pattern expressed by the cell types in the finalised phenotype i.e. at the last available development step.

The target pattern is a chessboard. To represent the chessboard only two types of cells are required i.e. cell type is expressed as black or white. The development process starts from a single cell and develops to an organism of size given by the predefined maximum phenotype size. In the initial condition for development all cells but the first single cell is defined as empty.

The chessboard pattern is highly scalable and can provide a good foundation for exploration of the evolution and development processes.

In this experiment the main scaling parameter investigated is the phenotype size i.e. organism. As such, the number of available cells in the phenotype is scaled while keeping the genotype size constant. The experiments are repeated for an increasing genotype size as an attempt to relate the results to the genome scaling for evolution of functional properties presented in section 4.

The development process used is the development model presented in section 2. Even though a hardware implementation is available the experiments in section 3 are carried out on a software platform. The experiments only take advantage of the cell types. The functional components of the cell in Figure 1 are not implemented. However, the cells express their type as an intercellular property. In an evaluation only targeting structural properties the cell type itself is enough to express a cellular building block in the phenotype structure.

3.1 Evolutionary Algorithm

The evolutionary strategy chosen incorporates survivor selection with elitism. Only cloning and mutation are used as genetic operators. The population size is constant and only the best individual is cloned to next generation. If there are two or more best individuals equally fit the newest one is preferred. The rest of the individuals are deleted and the new population is filled with mutants of the best. All individuals are re-evaluated in each generation.

If an individual is mutated the mutation operator can change one, two or three genes in the genome. The mutation points are chosen randomly. A gene at a mutation point is assigned a new random value. This might result in no change.

Table 1: Defined Cell types.

Cell Type	Cell name	Graphical representation
B	*Black*	●
W	*White*	○
Z	*Empty*	●

Table 1 list the available cells and their graphical representation used herein. The only cell types available are the empty, black and white. In addition to the given cell types the growth directions and DC condition are valid gene types in a rule.

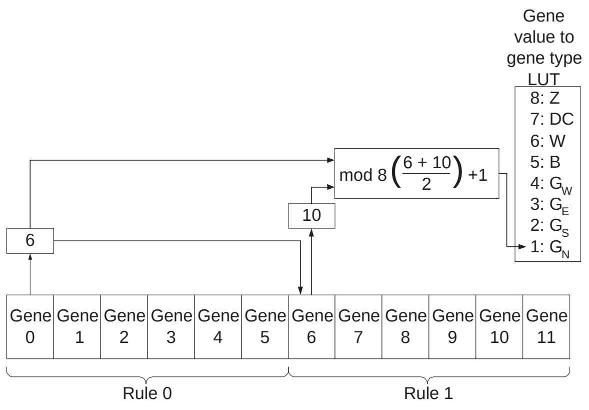


Figure 4: Mapping of gene values to gene types.

The genome is a symbolic representation of positive integers. Each rule is indirectly represented by six genes in sequence. The representation of each gene type i.e. B, W, Z, DC and growth direction, is not direct. Instead each gene value represents an address to another gene in the genome.

Figure 4 illustrates the process of mapping the positive integer gene representation in the EA to the gene type representation used in the development genomes. In the example a two rule genome is shown. The integer representation represent each gene by a value from 0 to the number of available genes. Here 0 to 11. Each gene value is mapped to a gene type in the following manner: The gene value, here 6 for the first gene, points to gene number 6. Gene number 6 has the gene value 10. The gene type for the first gene is found by adding 1 to mod 8 of the average of the two gene values. The result is used as an address to point in to a gene value to gene type LUT. As shown the gene type for the first gene in the first rule is a growth condition (Gn).

The genotype representation used in the EA have a neutral mapping [5] to genomes that can be processed by the development process described in section 2. The selection of the newest individual if equally high fitness is obtained can exploit neutral jumps in the phenotype space [5].

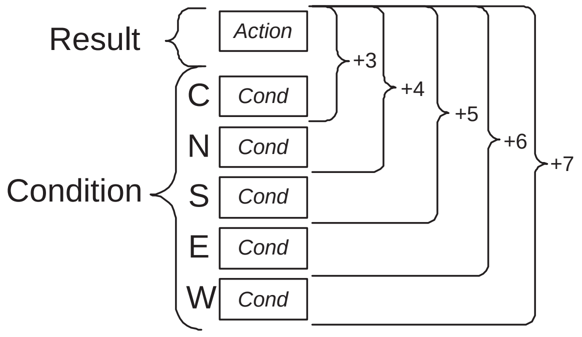


Figure 5: Fitness points based on correct composition of genes in a rule.

3.2 Evaluation

As stated in section 1 evaluation in a development process depends on the interplay between the processing of the genome in the cells and the emerging phenotype. As such, the development process require genomes capable of growth and differentiation to provide feedback to the EA. As an attempt to provide genomes containing valid rules to the development process valid gene entry in rules are rewarded — a rule fitness. To avoid waste of computation time only rule fitness is calculated for genomes containing no valid rules i.e. genomes are not processed by the development process only rule fitness points are given.

The rule fitness is a function based on three properties of each rule in the genome. First, each gene is given a reward independent of rule composition. That is, if a gene entry is valid i.e. no DC in the action and no growth entry in the conditional part. Second, the rule is given an increasing number of points based on the correctness of the rule composition i.e. the restrictions for growth and change rules. Third, a valid rule is given a reward.

The three different rule fitness properties are rewarded and weighted. Each correct gene is given 2 points. The calculation for correctness in rule composition is shown in Figure 5. If the action part and centre (C) condition is correct for a change or growth rule 3 points are given. If the action, centre and north (N) condition is fulfilled 4 more points are given, etc. A total of 25 points if all gene entries are correct. Each rule in compliance with all restrictions is given 20 points. Note that rule fitness value depends on genome size.

Genomes including valid rules are developed and given a

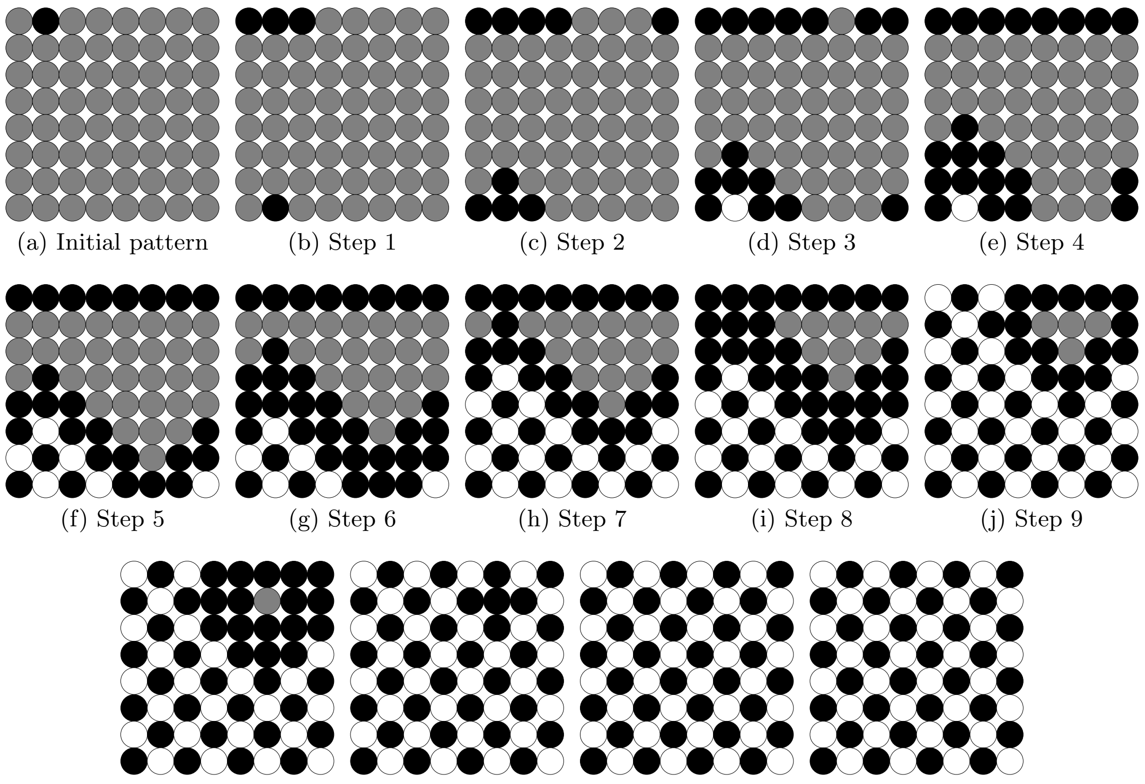

(a) Initial pattern (b) Step 1 (c) Step 2 (d) Step 3 (e) Step 4 (f) Step 5 (g) Step 6 (h) Step 7 (i) Step 8 (j) Step 9 (k) Step 10 (l) Step 11 (m) Step 12 (n) Step 13

Figure 6: Development steps for final individual. The pattern is stable from step 12 until last step, step 20. Grey cells are empty(Z).

fitness value based on the number of correct cell types in the finalised developed phenotype i.e. on the last available development step. Each structural correct cell type i.e. black or white is given 50 points.

The development example in Figure 6 shows the steps for the best individual after 3000 generations. This example develops from single black cell to a multicellular organism of 8 by 8 cells expressing the desired chessboard pattern. The example shows a perfect solution. The phenotype is structural stable at development step 12 i.e. no change in the phenotype from development step 12.

The organism in Figure 6 obtained a fitness of 3200 for it's structural composition. A rule fitness of 286 in total where 100 was rewarded for the 5 correct rules. The individual is rewarded a total fitness value of 3486.

Table 2: Genome for phenotype in Figure 6. Rule 4 have highest priority.

Rule	Result [Action]	Center [Cond]	North [Cond]	South [Cond]	East [Cond]	West [Cond]
4	*Gw*	*Z*	*DC*	*Z*	*Z*	*DC*
3	*Ge*	*Z*	*DC*	*Z*	*DC*	*DC*
2	*Gs*	*Z*	*DC*	*DC*	*DC*	*DC*
1	*W*	*W*	*W*	*Z*	*Z*	*DC*
0	*B*	*B*	*B*	*B*	*B*	*B*

The genome for the development process shown in Figure 6 is presented in Table 2. The five exploited rules consists of three growth rules and two change rules.

3.3 Experiment and Results

To investigate how phenotype size influence on the result produced by the EA the number of cells available to development was increased. Four different runs were conducted for phenotypes of size 16, 64, 256 and 1024 cells. The genotype size was set to 4 rules.

The initial condition was set to a single cell of type black (the zygote). The number of available development steps was set to 100. The population size is set to 5. The maximum number of available generations is 3000.

Table 3: Results of rule size 4: phenotype size scaled from 16 to 1024 cells.

Rules	Size	SR
4	16	33%
4	64	41%
4	256	34%
4	1024	44%

Table 3 shows the results of the four runs increasing the maximum size of the phenotype. In the table *Rules* gives the number of possible rules in the genome. *Size* is the maximum number of cells available for development i.e. organism size. The Success Ratio (*SR*) gives the percentage of perfect solutions found over 100 repeated runs.

In Table 4 the number of rules is increased to 5 rules. The

Table 4: Results of rule size 5: phenotype size scaled from 16 to 1024 cells.

Rules	Size	SR
5	16	55%
5	64	64%
5	256	59%
5	1024	61%

size of the phenotype is increased from 16 to 1024 cells. The results of the runs are presented in the table.

Table 5: Results of rule size 6: phenotype size scaled from 16 to 1024 cells.

Rules	Size	SR
6	16	68%
6	64	70%
6	256	73%
6	1024	66%

Finally in Table 5 the number of rules is increased to 6. The size of the phenotype is again increased from 16 to 1024 cells as for the two previous experiments. The results are presented in the table.

The gene activation pattern for development of one of the 5 rule genotypes that produced a perfect solution at the final development step is presented in Figure 7. Figure 6 shows the development of the phenotype. The plot in Figure 7 illustrates the gene activation together with the number of active rules in the organism at each development step. Rule numbers from 0 to 4 are placed on the left Y-axis. The mark (+) in the plot indicates that the rule was activated at the given development step. The right Y-axis show the number of cells in the organism with a active rule on a given development step. The number of active rule cells at each development step is illustrated by the plotted line.

The plot has an increasing number of active rules from the first development step to a maximum of 16 at development step 6. From development step 6 the number of active rules decreases down to 0 at the development step 12. The phenotype has reached a state of structural stability.

4. GENOME SCALING

From the task of evolving genomes for development of organism with structural properties in the previous section the task is now changed to evolve genomes for development of organisms with functional properties.

Functionality is expressed by the output of the functional components of the cell shown in Figure 1. As such, state steps are required to measure functionality in the phenotype.

Evolution of static and sequential behaviour is defined in [23]. Herein static behaviour is the target behaviour. Static behaviour is expressed as a global property of all cells included in the phenotype at a given state step at a given development step. The global properties chosen is the number of cells outputting a logical ”1”. As such, the GA search for genomes that can produce a phenotype consisting of suitable cell types at a given development step producing a desired

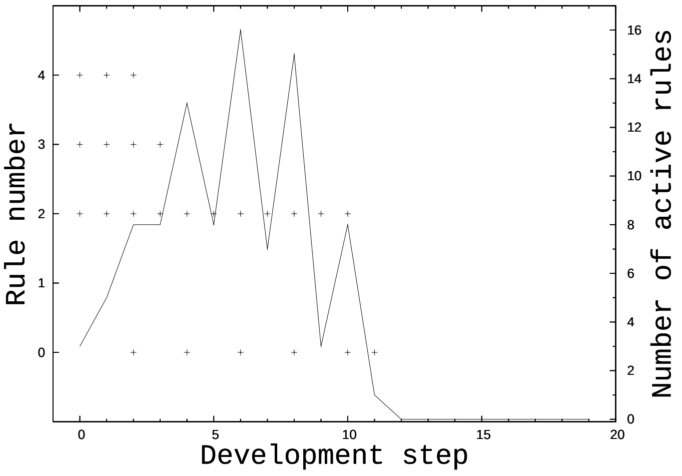


Figure 7: Gene activation pattern showing the rule activity in the development of the phenotype in Figure 6.

state output at a given state step.

To be able to produce an organism able to output the desired function of 1024 logical ”1”s, the genome must be capable of development of a structure that exploits all available cells. This structural requirement is not given in the fitness function but is a result of the fact that empty cells must be occupied to change their output value. As such, the functional goal can not be achieved if the phenotype structures do not exploit all available cells.

The cellular array used consists of 1024 cells –an array of 32 by 32 cells. The genome size is scaled by increasing the number of available rules that can be exploited by the development process.

The experiments are executed on a cPCI machine including a cPCI PC running the GA. The genomes are transferred on the cPCI bus to the FPGA card. The development process and functional behaviour of the cellular array are executed in the FPGA. Information from the developing phenotype is available by readback to the GA.

4.1 Evolutionary Algorithm

The approach here is to evolve genomes that can produce an emerging phenotype with functional properties. A fixed number of cell types to be exploited by evolution is chosen. The number of development steps and state steps is set to be fixed. As such, evolution must be able to find a solution within the given parameters. The set of cell types used are given in Table 6 together with their functional LUT definition and graphical symbols for graphical presentation.

The EA chosen is a straight forward Genetic Algorithm (GA) [22]. The GA’s crossover operator for multiple crossover points is modified. The modification has two purposes. First, to avoid the context of genes to be disturbed, i.e. genes representing a cell type or growth direction is copied unmodified by the crossover operator. Second, the number of crossover points is variable within a given range. The variable number of points is implemented to be able to move rules up and down in the ranking, i.e. placement within the genome.

The representation of cell types and growth directions are not represented as symbols. Each gene is represented by a

Table 6: Defined Cell types and their functionality

Cell Type	LUT hex	Function name	Graphical representation
0	$0xFFFF0000$	$Empty$	●
1	$0xFFFFFFFE$	$T \geq 1$	●
2	$0xFFFEFEE8$	$T \geq 2$	●
3	$0xFEE8E880$	$T \geq 3$	●
4	$0xE8808000$	$T \geq 4$	●
5	$0x80000000$	$T \geq 5$	●
6	$0x96696996$	$XOR5$	●
7	$0x00000001$	$\overline{T \geq 1}$	●
8	$0x00010117$	$\overline{T \geq 2}$	●
9	$0x0117177F$	$\overline{T \geq 3}$	●
10	$0x177F7FFF$	$\overline{T \geq 4}$	○
11	$0x7FFFFFFF$	$\overline{T \geq 5}$	●
12	$0x69969669$	$\overline{XOR5}$	●

given number of bits. Seventeen bits are used to represent the twelve used cell types in Table 6 together with the four possible growth directions and the don't care condition. The gene value is the number (sum) of bits set to logical "1" in a gene.

Further, the gene value does not directly reflect the gene type i.e. cell type, growth direction or condition. Instead of a direct translation of gene values to gene types a look-up table is used to translate the gene value to a gene type. The translation table used is given:
$9, 15, 2, 3, 4, 5, 6, 17, 14, 7, 16, 17, 8, 0, 13, 11, 12, 10$

A gene value of zero point to the first entry in the look-up table defined as cell type 9 in Table 6. The entries 13 – 16 represents the four growth directions and 17 the don't care condition. Using a look-up table for gene translation has two important properties. First, it is easy to remove cell types from the available cells as cell types not represented in the table can not be expressed in the phenotype. Second, the table can be exploited by the genetic operators. A single bit mutation in a gene will change the gene type one step up or down the table.

The purpose of the uneven distribution of gene values to gene types caused by the gene value representation is to keep the experiments herein compatible with ongoing work.

4.2 Evaluation

The initial condition is applied before development starts. This means that all cells are set or reset depending on the given initial condition. To avoid empty cells updating their output values from their Von Neumann neighbourhood, all cells of type Empty are set to update their outputs based on only their own output value at the previous clock pulse. As such, a given empty cell will retain its initial state until the emerging organism grows into it.

In contrast to the experiments in section 3, where a part of the fitness function is defined to reward rules that are composed correctly, here the fitness function rewards genomes for including rules that is exploited during development.

In section 3 the argument for rewarding correct composition within a rule was to favour such genomes to be able to exploit the genetic material towards genomes that was capable of growth and differentiation i.e. a zygote that starts to develop. Here a different approach is taken. All genomes in the initial population are made of gene combinations that are valid rules. However, there are no mechanism preventing the genetic operators to produce invalid rules.

A fitness function that included activated rules in the emerging phenotype together with the number of cells outputting a logical "1" at the final state step at the final development step was defined. The fitness function is the sum of activated rules multiplied by 2 added to the number of cells outputting a logical "1". As such, the number of cells outputting a logical "1" defines the success of the structure developed. Exploited rules keep useful exploitable genetic material in the population.

The development example in figure 8 shows the development of an organism of 32 by 32 cells over 50 development steps. In the example a single cell of type XOR5 outputting a logical "1" (the zygote) develops to a multicellular organism of 1024 cells. The target for evolution is to produce organisms outputting as many logical "1"s as possible i.e. 1024. The finalised organism at the last available development step is a structure of cell type 8 ($\overline{T \geq 2}$) and 11 ($\overline{T \geq 5}$). A genome size of 32 rules was used. Out of the available rules 6 rules was exploited. The presented emerging phenotype is a result of 8619 generation of evolution.

The functional property of the phenotype was given a score of 1024 points based on the number of logical "1" outputted on the last state step of the last available development step. The rule fitness was given 12 points based on the 6 exploited rules out of the available 32 rules in the genome. A fitness of 1038 points.

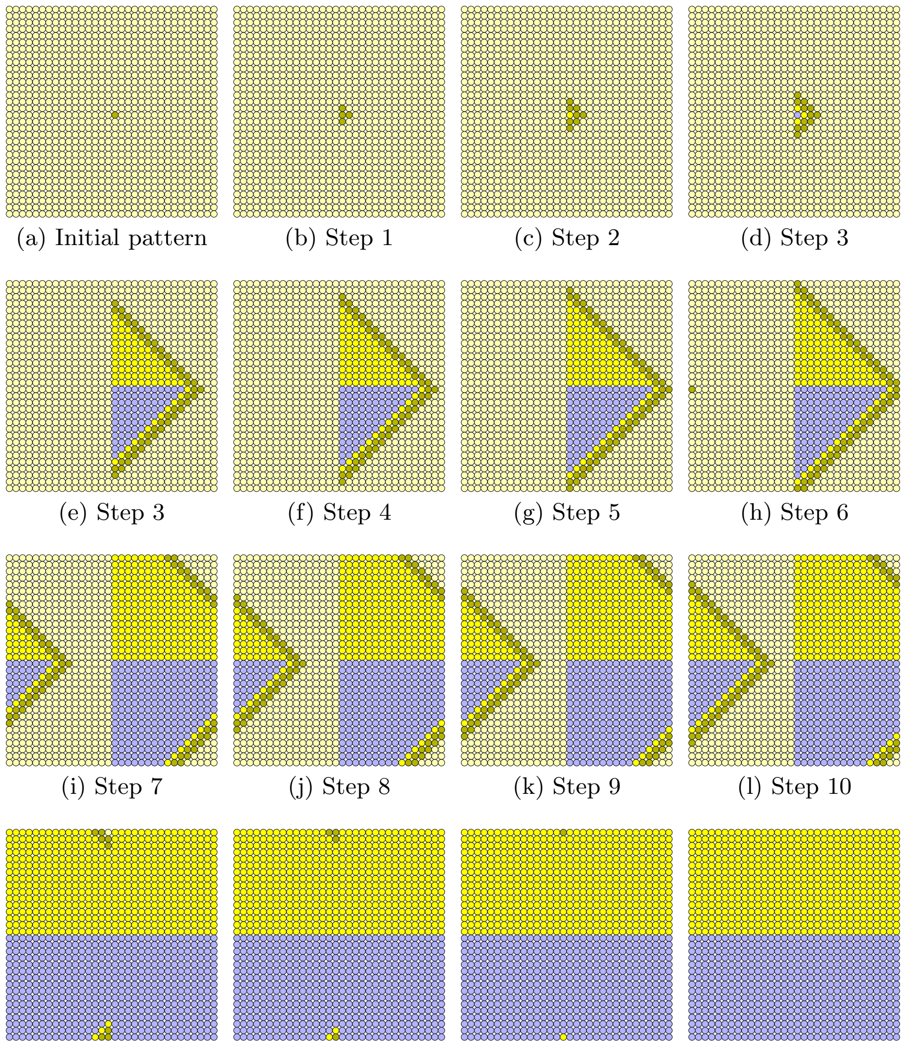
Table 7: Genome for phenotype in Figure 8. Rule 31 have highest priority.

Rule	Result [Action]	Center [Cond]	North [Cond]	South [Cond]	East [Cond]	West [Cond]
0	Gs	DC	DC	Gs	3	DC
1	DC	DC	Gs	Gs	DC	DC
2	DC	DC	Gs	Gs	6	DC
3	Gs	0	6	Gw	Gs	DC
4	8	6	DC	DC	DC	8
5	DC	DC	5	Gs	8	Gw
6	6	5	Gs	Gs	Gw	Gs
7	DC	DC	DC	Gw	8	DC
8	8	6	DC	DC	6	DC
9	DC	Gw	5	6	6	Gs
10	DC	DC	6	Gw	Gw	4
11	DC	6	5	DC	DC	DC
12	DC	6	Gw	Gs	DC	0
13	DC	6	DC	Gs	DC	8
14	DC	Gw	Gs	DC	5	Gw
15	DC	Gw	6	6	Gw	Gs
16	Gn	0	DC	DC	DC	DC
17	Gs	0	6	DC	6	5
18	Gw	0	DC	DC	DC	DC
19	DC	Gs	DC	Ge	6	5
20	DC	DC	Gw	5	4	DC
21	6	DC	8	DC	Gs	Gs
22	Gs	0	DC	DC	DC	DC
23	6	8	11	DC	5	DC
24	DC	Gw	Gs	0	Gs	DC
25	DC	Gs	Gw	DC	DC	Gs
26	8	DC	5	Gs	Gw	Gs
27	Gw	DC	0	Gs	DC	DC
28	Gw	0	Gw	Gs	Gs	0
29	Gs	0	Gs	DC	8	DC
30	Gs	0	5	Gw	DC	Gs
31	11	8	DC	6	DC	DC

The genome for the development process shown in Figure 8 is shown in Table 7. The 6 exploited rules consists of three growth rules and three change rules.

4.3 Experiment and Results

The initial state is set to be a single cell of type 6 (XOR5)



(a) Initial pattern (b) Step 1 (c) Step 2 (d) Step 3

(e) Step 3 (f) Step 4 (g) Step 5 (h) Step 6

(i) Step 7 (j) Step 8 (k) Step 9 (l) Step 10

(m) Step 11 (n) Step 12 (o) Step 13 (p) Step 49

Figure 8: Selected development steps for development of static behaviour. The output states of the cells makes a bit array of 1024 cells outputting a logical "1".

the cell is set to output a logical "1". The population size is set to 16. The genome size was scaled from 4 to 32 rules. Development steps were set to 50. State steps for each development step was set to 30. The number of activated rules and the cell types in the cellular array was recorded together with the fitness value.

Note that due to it's functional properties of once set always producing a logical "1" the cell type $T \geq 1$ is not included in the runs.

Table 8: Result of searching for static behaviour.

Rules	Used rules best/mean	"1"s best	Fitness best/mean	Generation best/mean
4	4/3.4	909	917/438.2	10721/9222,6
8	6/4.8	920	932/744.7	8816/10731.4
16	7/5.5	1024	1038/931.9	17685/10000.3
32	8/6.8	1024	1040/1010.4	15565/10057.6

The result can be found in Table 8. The table contains the 4 different runs. Each run was repeated 10 times. For each run the number of active rules, number of cells outputting a logical "1" and the fitness score for the best and mean over the 10 runs is presented. In the last column the respective generation for the best individual is presented together with the mean generation for the best of 10 runs.

The gene activation pattern for development of one of the 32 rule genotypes that produced a perfect solution of outputting 1024 logical "1"s at the final development step is presented in Figure 9. Figure 8 shows the development of the phenotype. The plot in Figure 9 illustrates the gene activation together with the number of active rules in the organism at each development step. Rule numbers from 0 to 31 are placed on the left Y-axis. The mark (+) in the plot indicates that the rule was activated at the given development step. The right Y-axis show the number of cells in the organism with an active rule on a given development step.

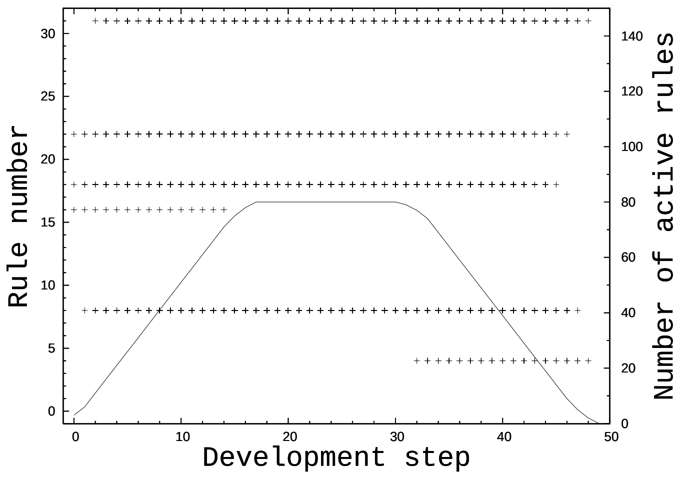


Figure 9: Gene activation pattern showing the rule activity in the development of the phenotype in Figure 8.

The number of cells with active rules at each development step is illustrated by the plotted line.

The plot has an increasing number of active rules from the first development step to a maximum of 80 at development step 17. From development step 17 to 30 the number of active rules is constant at 80 active rules. At development step 32 the number of active rules decreases down to 0 at the last development step. The phenotype has reached a state of structural stability. Note that not all of the produced phenotypes in the experiment reached structural stability.

5. DISCUSSION

In the experiments considering phenotype scaling the structure emerges as a result of the interplay between the genome and the phenotype. The goal structure may be said not to be a true emerging global structure. Each cell can take action to express it's correct cell type only by examining the local neighbourhood. However, the genome presented to the development process must be arranged in such a way that it is able to develop the target pattern.

The true parallel cellular development process require gene regulation. Regulating the genes to grow and differentiate without constructing structural conditions that is not covered in the limited size of the genome or leads to oscillating substructures. In the parallel cellular development process all cells execute the development process in parallel. Cells can only rely on the information provided by it's neighbours at the current development step. As such, a cells action based on the structural information provided is unambiguous regarding the cell's development action i.e. change or growth.

The result of evolution of developmental genomes for solving the cellular structure of a chess board indicate that the ES can almost keep it's performance or in some cases take advantage of the increased number of available cells to increase the success ratio.

The increased success ratio if the number of available rules was increased also show that the ES was able to effective exploit the available gene regulation network.

In the experiments for genome scaling a true global property that can only be achieved as an emerging property out of local interactions is the target. The genome and the development process must construct a structure of functional components that can meet the target criteria. The removal of the cell type $T \geq 1$ ensure that a solution exploit the functional components of the cells using the state steps. There are no cells avilable that can be used by growth and differentiation alone to build a structure of cells outputting a logical "1".

The result of the experiment shows that evolution was able to exploit the increased genome size toward better solutions. However, it is clear that only parts of the genome are exploited. Looking at the gene activation plot in Figure 9 for the organism in Figure 8 only six of the rules are exploited. However, the genome carry information from the evolutionary process that is not exploited in the current generation.

Looking at the graphical presentation of development in Figure 8 shows that the final phenotype is constructed of two cell types only. However, the phenotype is developed exploiting a third cell type in the growth and differentiation process. This third cell is part of the actual result obtained even if it is not present at the final development step. The cell is part of the functionality expressed. The final output bit pattern is a result of all development steps and state steps from the first cell outputting a logical "1".

If the presented example of gene regulation plots in Figure 7 and Figure 9 and graphical presentation of the developing organisms in Figure 6 and Figure 8 are examined both EAs found stable structural phenotypes at the last development step available. There are no specific mechanisms for self regulation available. Evolution is capable of finding genomes that develop to organisms were self-regulation emerges out of the local cell interactions.

In the experiments considering phenotype scaling and experiments for genome scaling a common obstacle must be solved. The number of available development steps was set to be fixed. As such, the structural requirement in both cases of exploiting all available cells the developmental genome must provide a growth ratio that can expand the organism in the appointed number of development steps.

6. CONCLUSIONS

The experiments regarding phenotype scaling presented in section 3 and genome scaling in section 4 indicate that increased number of available cells in the phenotype and the size of the genome influence the results in a positive direction.

The fact that both EAs were able to exploit the development model was encouraging. This may indicate that a development mapping may not be that sensitive to the EA implementation and parameter tuning.

The increased performance in the experiments by the increased size of the genome and to some extent the phenotype size indicates that the development process is capable of exploiting the increased information available. Information here is an increased number of structural chess board cells that can be used in the developmental computation to construct the finalized board. The increast number of available rules leading to a larger number of instructions also provide an increast amount of information i.e. rules, that can be exploited during the development process.

7. REFERENCES

[1] K. Aamot. Kunstig utvikling: Utvidelse av fpga-basert sblock-plattform. Master's thesis, The University of Science and technology, Norway, 2005.

[2] P. Bentley and S. Kunar. Three ways to grow designs: A comparison of embryongenies for an evolutionary design problem. In *GECCO*, pages 35–43, 2000.

[3] P. J. Bentley, editor. *Evolutionary Design by Computers*. Morgan Kaufmann Publishers,Inc, San Francisco, USA, 1999.

[4] S. Droste, T. Jansen, G. Rudolph, H. Schwefel, K. Tinnefeld, and I. Wegner. Theory of evolutionary algorithems and genetic programming. In H. P. Schwefel, I. Wegener, and K. Weinert, editors, *Advancec in Computational Intelligence Theory and Practice*, pages 107–144. Springer, 2003.

[5] M. Ebner, M. Shackleton, and R. Shipman. How neutral networks influence evolvability. *Complexity*, 7(2):19–33, 2001.

[6] T. G. W. Gordon. Exploring models of development for evolutionary circuit design. In *2003 Congress on Evolutionary Computation (CEC 2003)*, pages 2050–2057. IEEE, 2003.

[7] T. G. W. Gordon and P. J. Bentley. Bias and scalability in evolutionary development. In *the 2005 Genetic and Evolutionary Computation Conference*, pages 83 – 90. ACM Press, 2005.

[8] T. G. W. Gordon and P. J. Bentley. Development brings scalability to hardware evolution. In *the 2005 NASA/DOD Conference on Evolvable Hardware (EH05)*, pages 272 –279. IEEE, 2005.

[9] P. Haddow and G. Tufte. An evolvable hardware FPGA for adaptive hardware. In *Congress on Evolutionary Computation(CEC00)*, pages 553–560, 2000.

[10] S. Kumar. *Investigating Computational Models of Development for the Construction of Shape and Form*. PhD thesis, University College London (UCL), 2004.

[11] S. Kumar and P. J. Bentley. Biologically inspired evolutionary development. In *5th International Conference on Evolvable Systems (ICES03)*, Lecture Notes in Computer Science, pages 57–68. Springer, 2003.

[12] S. Kumar and P. J. Bentley, editors. *On Growth, Form and Computers*. Elsevier Limited Oxford UK, 2003.

[13] J. Liu, H.and Miller and A. Tyrrell. Intrinsic evolvable hardware implementation of a robust biological development model for digital systems. In *the 2005 NASA/DOD Conference on Evolvable Hardware (EH05)*, pages 87 – 92. IEEE, 2005.

[14] D. Mange, S. Moshe, A. Stauffer, and G. Tempesti. Towards robust integrated circuits: The embryonics approach. *Proceedings of the IEEE*, 88(4):516–543, April 2000.

[15] D. Mange, E. Sanchez, A. Stauffer, G. Tempesti, P. Marchal, and C. Piruet. Embryonics: A new methodology for designing field-programmable gate array with self-repair and self-replicating properties. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 6(3):387–399, September 1998.

[16] J. Miller and K. Downing. Evolution in materio: Looking beyond the silicon box. In *2002 NASA/DOD Conference on Evolvable Hardware*, pages 167–176. IEEE Computer Society Press, 2002.

[17] J. F. Miller. "evolving developmental programs for adaptation, morphogenesis, and self-repair. In *Seventh European Conference on Artificial Life*, Lecture Notes in Artificial Intelligence, pages 256–265. Springer, 2003.

[18] J. F. Miller and P. Thomson. A developmental method for growing graphs and circuits. In *5th International Conference on Evolvable Systems (ICES03)*, Lecture Notes in Computer Science, pages 93–104. Springer, 2003.

[19] E. Sanchez, D. Mange, M. Sipper, M. Tomassini, A. Prez-Uribe, and A. Stauffer. Phylogeny, ontogeny, and epigenesis: Three sources of biological inspiration for softening hardware. In T. Higuchi, M. Iwata, and W. Liu, editors, *Evolvable Systems: from Biology to Hardware, ICES 96*, volume 1259 of *Lecture Notes in Computer Science*, pages 35–54. Springer, 1996.

[20] M. Sipper. *Evolution of Parallel Cellular Machines The Cellular Programming Approach*. Springer-Verlag, 1997.

[21] M. Sipper. The emergence of cellular computing. *Computer*, 32(7):18–26, 1999.

[22] W. M. Spears. Gac ga archives source code collection webpage. `http://www.aic.nrl.navy.mil/galist/src/`, 1991.

[23] G. Tufte. Cellular development: A search for functionality. In *Congress on Evolutionary Computation(CEC2006)*. IEEE, 2006.

[24] G. Tufte. Gene regulation mechanisms introduced in the evaluation criteria for a hardware cellular development system. In *1st NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2006)*. IEEE, 2006.

[25] G. Tufte and P. Haddow. Biologically-inspired: A rule-based self-reconfiguration of a virtex chip. In *4th International Conference on Computational Science 2004 (ICCS 2004)*, Lecture Notes in Computer Science, pages 1249–1256. Springer, 2004.

[26] G. Tufte and P. C. Haddow. Building knowledg into developmental rules for circuite design. In *5th International Conference on Evolvable Systems (ICES03)*, Lecture Notes in Computer Science, pages 69–80. Springer, 2003.

[27] G. Tufte and P. C. Haddow. Towards development on a silicon-based cellular computation machine. *Natural Computation*, 4(4):387–416, 2005.

[28] A. Tyrell, E. Sanchez, D. Floreano, G. Tempestti, D. Mange, J. Moreno, J. Rosenberg, and A. E. P. Villa. Poetic tissue: An integrated archtecture for bio-inspired hardware. In *5th International Conference on Evolvable Systems (ICES03)*, Lecture Notes in Computer Science, pages 127–140. Springer, 2003.

[29] J. Von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, Urbana, IL, USA, 1966., 1966.

[30] L. Wolpert. *Principles of Development, Second edition*. Oxford University Press, 2002.