

# Phenotypic, Developmental and Computational Resources: Scaling in Artificial Development

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

## ABSTRACT

Developmental systems have inherent properties favourable for scaling. The possibility to generate very large scale structures combined with gene regulation opens for systems where the genome size do not reflect the size and complexity of the phenotype. Despite the presence of scalability in nature there is limited knowledge of what makes a developmental mapping scalable. As such, there is few artificial system that show true scaling. Scaling for any system, biological or artificial, is a question of resources. Toward an understanding of the challenges of scalability the issue of scaling is investigated in an aspect of resources within the developmental model itself. The resources are decomposed into domains that can be scaled separately each may influence on the outcome of development. Knowledge of the domains influence on scaling provide insight in scaling limitation and what target problems that can be scaled. The resources are decomposed into three domains; Phenotypic, Developmental and Computational (PDC). The domains are placed along three axes in a PDC-space. To illustrate the principles of scaling in a PDC-space an experimental approach is taken.

## Categories and Subject Descriptors

I.2.2 [Artificial Intelligence]: [Cellular arrays and automata]

## General Terms

Design, Experimentation

## Keywords

Development, Cellular Computation, Evolvable Hardware

## 1. INTRODUCTION

One of the main topics of interest in introducing mapping processes inspired by development [34] to Evolutionary

Algorithms (EAs) is the aspect of scalability. In the work of Kitano [20] the scaling issue was addressed by searching for an effective encoding in order to generate more complex network structures. The urge towards evolution of complex network structures was motivated by a need to handle scaling of the complexity of solvable problems.

The proposed solution was a developmental system i.e. an indirect mapping. In contrast, in a direct mapping approach there is a single phenotype, defined by the genotype. This implies that each element in the phenotype is represented explicitly in the genotype. As such, introducing mechanisms, inspired by development, into evolutionary computation was motivated by a need to overcome limitations in such direct mapping approaches, e.g. scaling.

A developmental mapping is an example of an indirect mapping. In biological development, an initial unit — a cell, holds the complete building plan (DNA) for an organism. It is important to note that this plan is generative — it describes how to build the system, not what the system will look like. Similarly in a developmental mapping, the artificial organism starts out as a single cell where the genome provides the cell's DNA. The processing of the genome may be based on gene regulation. Each development step, or stage in the mapping, produces a candidate phenotype which continues to emerge during development. Gene regulation implies that different parts of the genome are expressed in different cells at different times in the emerging phenotype. The organism emerges as a result of an interplay between the genome and the intermediate phenotypes e.g. a cell's next action may depend on the DNA (genome) and it's neighbours current states (intermediate phenotypes). The environment may also affect the phenotype emerging from the development process. That is, the genome develops in an environment where the emerging phenotype is tuned by the environment in which it develops.

Taking inspiration from nature into artificial developmental systems may be motivated by a desire to include inherent properties favourable for scaling. The generative process can generate very large scale repetitive structures [21]. Gene regulation opens for a genome size that may not reflect size or complexity of the developed phenotype. As such, a developmental system can create structure of arbitrary size for problems that can be solved by increasing the number of repeated structures. The relative small genome size may decrease or change the search space by being regulated by the emerging organism and the environment thus increasing evolvability [19].

Scaling for any system, biological or artificial, is a question

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

GECCO '08, July 12–16, 2008, Atlanta, Georgia, USA.  
Copyright 2008 ACM 978-1-60558-130-9/08/07...\$5.00.

of resources. As such, there is a question of what resources those are available to the system. To be able to deal with such questions knowledge of what domain the resource scaling fit in with is necessary. Giving two different, however not independent, resource domains for scaling. First, scaling of mapping external factors. This domain include number of generations, population size, genome size and other factors relating directly to the EA, here termed extrinsic resources. The second domain relates to scaling in relation to the development process itself e.g. phenotype size, cellular actions and information processing. Here termed intrinsic resources. The latter is the main topic pursued in this work and covered in detail in Section 3.

The intrinsic resources are partitioned along three axes: Phenotypic, Developmental and Computational; referred to as PDC-resources. To illustrate the principles of scaling developmental resources an experimental approach is taken.

The article is laid out as follows: background information and selected related works are presented in Section 2. Section 3 introduces the PDC-space. In Section 4 the development model used in the experiments is presented together with the EA. Experiments showing scaling within the PDC-space are presented in Section 5. Finally Section 6 concludes the work.

## 2. BACKGROUND

The issue of scalability in developmental systems have lead to several approaches. In the different approaches diverse scaling factors/properties have been investigated.

Bentley and Kumar [4] addressed the scalability by investigate the performance of direct encoding versus developmental encodings to generate phenotypes of different structural size, e.g. number of cells in the final phenotype. Other similar approaches dealing with phenotypic size as the scaling factor have looked into the affect on performance of the development process regarding scaling of phenotypic size and genome size [33].

Another way of considering scaling is to look at the functional property of the developed phenotype [21]. In such approaches there has been work on keeping-up functional performance when the size, i.e. number of neurons, of the phenotype increases [11] or the size of the genome, i.e. search space size, increases [29].

Even if the examples of scaling in developmental systems were successful within the specified criteria for the experiments, the goal of increasing the complexity of solvable problems is not truly shown. The development of check boards of different size in [33] emphasise this statement. If the size of the sought target check board was scaled-up the performance of the system remained the same, or even improved, this is due to the fact noted in [21]; the development process can generate very large scale repetitive and structured systems. As such, it is not given that it is a harder problem for a developmental system to create a large check board then a small.

Gordon [15] showed a successful attempt to scale-up the number of bits in an adder circuit from two to seven. However as noted by Gordon development exploited the underlying technology for the specific problem size. The development system did not scale by creating a repetitive adder structures that are truly scalable and known from literature on digital design [10]. Despite, the examples of successful scaling of target structures [4] and functionality [15], the re-

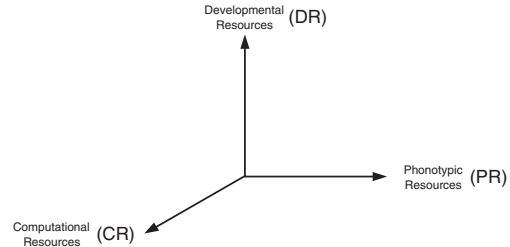
sults only show scaling within the very specific experimental setup, there is no general result showing a system that can scale with the complexity and problem size of the target behaviour.

Examples of more general solutions to scaling in developmental systems have been found. Sekanina and Bidlo [24] demonstrated development of arbitrarily large sorting networks. Harding et.al. [18] attacked the problem of finding a general solution that generates sequences of squares [28]. However, again these examples show scaling within a specific problem, i.e. not systems that scales toward more complex problems.

The diverse approaches and views described on scaling in developmental systems calls for an understanding of what the goal of scaling is, what changes to be made to scale, e.g. change in resources, and what results of scaling that can be expected for the problem at hand. These questions have lead to an attempt to classify resources present in a developmental model.

## 3. PHENOTYPIC, DEVELOPMENTAL AND COMPUTATIONAL RESOURCES

As stated in Section 1 developmental systems have inherent properties pointing towards a possible scalable system. The obvious example is the possibility to scale by exploiting developmental mechanisms to create a large structure of similar sub-structures. However, it is not obvious what factors that should be scaled in such approaches. The other example of scalability aiming to evolve and develop solution that scale with the complexity of the problem faces the same challenge of what factor to scale to achieve the goal sought.



**Figure 1: The tree axes of Phenotypic, Developmental and Computational resources.**

To better understand scaling of developmental systems and find a paradigm that can be used to connect scaling factors, scaling based on resources is proposed. The scaling taxonomy is based on a cellular developmental model – see Section 4.1. This does not imply that the resource view is not valid for other models. Other development models may be fitted within the scaling taxonomy by inserting relevant resource measurements.

The scaling of resources can be placed in a resource space given by the three axes: Phenotypic Resources (PR), Developmental Resources (DR) and Computational Resources (CR). In Figure 1 the resource space is shown as three axes each represent a scaling factor in the Phenotypic, Developmental and Computational Resource space (PDC-resources). Note that none of the axes have units. The resource unit for each axis must be considered for a specific development model and the target problem.

### 3.1 Phenotypic Resources

Resources directly relating to the size of the organism are placed along the X-axis in Figure 1. The size aspect relates to the available structural or functional resources for development of the phenotype, e.g. cells. It is important to note that the size may not reflect the final size of the developed phenotype. The size relates to available phenotypic resources for development. This is due to the fact that resources in the notion of cells may be exploited in phases of the development of an emerging organism, but killed or pruned in a later phase to achieve the goal sought. As such, development exploits the resources available, but the final phenotype may not exploit all available phenotypic resources. This effect can be observed in the work including phenotypic plasticity for environmental response [32] where parts of the final phenotype was pruned by the mechanism of cell death.

Considering a developmental model and a problem that would truly scale along the PR-axis (keeping the other resources constant). In such a case the problem size or complexity of solvable problems should be expanded if the development of the organism was given more physical resources. A physical resource in artificial development may refer to a hardware component [31] or more abstract a computer program representing an atomic unit of the organism [2].

### 3.2 Developmental Resources

Along the Y-axis are the resources responsible for the development of the emerging phenotype. This resource domain include the mechanisms that creates the phenotype, e.g. growth, differentiation, number of developmental iterations and other morphological mechanisms. In addition the information available, at any point in time, to the development process, e.g. inter-cell communication, environmental information or other information influencing the gene regulation, are included.

Scaling of information along the DR-axis influence what structures and what functionality that can be developed. von Neuman's Self-Reproducing Automata [5] was made possible by the definition of a given number of states a cell could hold, i.e. 29, and knowledge of the cell states in it's von Neumann neighbourhood, i.e. 5-neighbors. As such, self-reproduction was possible in a number of iterations with the defined inter-cell communication and a set of cell states. Later Codd [6] reduced the number of cell types to eight thus reducing the amount of information that can be expressed in the phenotype. The reduction in cell types in the two approaches scales the resources along the DR-axis.

In a developmental approach the available cell types that can be expressed in the phenotype may improve performance of the development process. This is demonstrated by Gordon [14] comparing a system with increased resources i.e. cell types and proteins, to Millers system [22] targeting development of French flag structures. The results indicated that the increased resources reduced the number of evaluations required. However, the resources in Millers system were sufficient for the task.

Increasing DR by extending the information available to the gene regulation can be considered as scaling up the developmental resources. Including environmental information [32] or adding chemicals to the development model may be a different way of scaling up the DR to tackle harder problems. However, information regarding cell types, inter-cell

communication, environment or chemicals increase the complexity of gene regulation by introducing more regulatory factors. As shown by Haddow [16] increasing information, i.e. chemicals, can decrease the performance of the system.

Scaling developmental resources regarding the aspect of cell growth, cellular differentiation and morphogenesis may be said to be a scaling of the information available for the gene regulation and in this manner setting bounds for what structures and functionality that might emerge. As such, the information available to the generative process of development and the possibility to express changes in the emerging structure are important for what organism that can be developed.

### 3.3 Computational Resources

The last resource along the Z-axis considers properties related to the behaviour of the basic units, e.g. a single cell, in the organism. The behaviour of the basic units contributes to the sought solution. As such, computational resources are the most problem specific domain in the PDC-space and relates mostly to systems with a target behaviour dealing with functionality i.e. some form of computation. Computation may not only relate to electronics or mathematics but also systems that can impose some form of change to itself or the environment [1]. Scaling along the CR-axis deals with the resources an organism have available toward fulfilling the evaluation criteria specified.

Systems explicitly including computational elements in the phenotype can obviously be scaled along the CR-axis. Such systems may include; Artificial Neural Networks (ANN) [2, 9], digital circuits [13] and cellular computation [31]. Computational resources may scale local connections for ANN [8], changing the possible logic expression in digital gates [12] or the number of iterations available to sequential digital circuits in a cellular computation paradigm [26]. The latter is investigated in section 5.

Other systems far from the traditional view on computation, i.e. a more general view of change, can be found in Eggenberger Hotz [7] where the cell division plane and the adhesion of each cell are exploited to impose change in the form of the phenotype. As such, the ability for expressing such cellular properties can be scaled along the CR-axis.

As stated the scaling of computational resources are the most application dependent domain. Scaling along this axis may change what functionality that may be developed. If the computational resources are expanded from e.g. the combinatorial circuits in [14] to include sequential circuits as in [31] functions requiring memory can be developed or contrariwise the cellular computational approach in [31] can develop functions including asynchrony behaviour. As such, scaling of computational resources can change the problem domain a developmental model can be applied to.

### 3.4 Scaling within the PDC-Space

The resources in the PDC-space are partitioned into domains however defining clear boundaries between the different domains is not trivial. If the number of cells is increased it may be said that the phenotypic and computational resources are expanded. However, as proven by Codd [6] increasing the number of cells, i.e. phenotypic resources, for a 2-state 5-neighbor CA is not sufficient to provide universal computation. On the other hand increasing the neighbourhood or states a cell can express, i.e. computational re-

sources, can provide the necessary resources [25]. Likewise, von Neuman’s Self-Reproducing Automata require a given number of cell types, i.e. developmental resources. As such, increasing the number of cells, i.e. phenotypic resources, are not sufficient. However, for all systems there must be a sufficient number of cells available, i.e. phenotypic resources.

In general, the scaling for a given development model and the applied problem have two challenges; First, the resources available must be able to handle the problem domain. This means that the development model must have the required developmental resources to express a structural phenotype with computational resources necessary for the functional requirements. Second, scaling of resources to achieve solution that can solve a larger problem size or a system that can solve a more complex problem.

## 4. EVOLUTION AND DEVELOPMENT

The development model used to illustrate scaling in the PDC-space is designed to target both structural and functional organisms. Here evolution is applied to tune the developmental genome for functional organisms. The phenotype may be evaluated at a given step of development, defined as the finalised phenotype, as in [15] or at each or any stage during development [31]. The latter takes the actual process of developing the emergent functionality [29] into the evaluation process i.e. life-time evaluation.

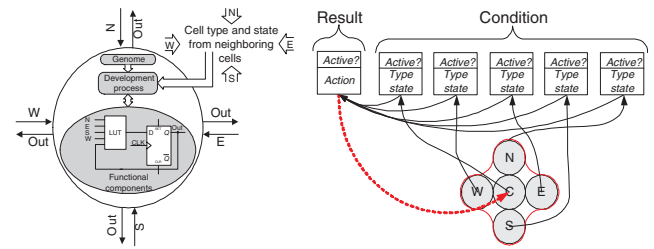
Targeting functionality with life-time evaluation implies that the effect of scaling within the PDC-space influence on the life-time functional performance of organisms. Evolution is applied to different cases with different PDC-resources available. As such, the influence of scaling the different resource domains within the development model can be monitored.

Herein the scaling issue is only investigated for intrinsic resources. Resources related to the extrinsic domain are kept constant. Scaling up extrinsic resources may influence on the results however this option is not further investigated in this work.

### 4.1 Development Model

The development model is based on cellular development. This implies that the genome is present and processed autonomously in every cell. In the model, the cell also contains the functional building blocks. For the experiments herein the application sought is that of a digital circuit (phenotype). Figure 2(a) illustrates the developmental system — 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 information in the functional components represents the type of the cell and the cell’s state is described by the outputs of the functional components.

The genome consists of a set of rules. Rules are restricted to expressions consisting of the type and state of the target cell and the types and state of the cells in its von Neumann neighbourhood. There are two types of rules i.e. change and growth rules. Cell growth is a mechanism to expand the organism. A growth rule result provides the direction of growth: grow from north **Gn**; east **Ge**; south **Gs** or west **Gw**. It is important to note that these rules are expressed in terms of where the source of the cell growing into the target cell is. Describing where a cell is growing from enables a fully parallel implementation of the system to be created



(a) Components of the cell. (b) Regulation of genes as interplay between the genome, cell types and the cell state of the environment or the emerging phenotype.

**Figure 2: The basic cell and a rule showing the gene regulation in the cellular development model.**

whilst retaining the possibility that cells in effect may grow in all four directions simultaneously. Growth rules have two restrictions. First, the target cell must be empty — this is to prevent growing over an existing cell and thus specialising the cell with a new cell type. Secondly, the cell to be copied into the target can not be empty.

Differentiation changes a cell’s type i.e. its functionality. The result part of a change rule states the type of cell the target is going to be changed into. Cells have the following types: valid cell types; don’t care (DC) or empty. However, the empty cell is not a valid target cell type.

Each rule consists of a result and a condition. The conditional part provides information about the cell itself and each of the neighbouring cells. To introduce external environment [32], state information is also needed. State information provides a way to include information relating to the functionality of the organism at a given point in time as well as information about the external environment — the empty cells in the environment also have state information. As such, a cell is represented in the condition of a rule by two genes representing its type and its state. However, a target cell is only represented by one gene: it’s type for change rules or growth direction for growth rules. The state of cell may be 0, 1 or DC. DC is introduced to provide the possibility to turn on or off this environmental influence.

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. Figure 2(b) illustrates the process of evaluating a rule. For each cell condition, the cell type and state are compared and if the conditions are 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 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 rule can not be activated. Activation of the second rule prevents activation of the third rule, etc.

The functional components of the cell is an Sblock [17]. The content of the look-up table (LUT) defines functionality and is, herein, also used to define the cell type. 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 and as a feedback to itself.

One update of the cell’s type under the execution of the development process is termed a development step (DS). A development step is thus 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). A development step is thus made up of a number of state steps.

The initial condition is applied before development starts. This means that all empty 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 — environmental information, until the emerging organism grows into it.

## 4.2 Evolutionary Algorithm

The evolutionary algorithm chosen is a Genetic Algorithm (GA), a modified version [29] of a straight forward GA found in [27]. The GA’s crossover operator is modified such that a gene is undisturbed and a variable number of crossover points is implemented.

## 4.3 Evaluation

The results presented in [31] illustrate a sequential counting based on the state information of the entire cellular space and the sequential operation of the functional components of the cells i.e. a cellular computation approach [26]. For the purpose of the experiments conducted herein, a counting behaviour is investigated thus placing a requirement on the tuning of the development genome (by evolution) and the emerging phenotype (by development) for such sequential digital circuit behaviour.

A counting sequence is defined in the cellular array as the number of logical "1"s in the cellular array increasing by one for each state step. As stated, an organism is evaluated over a given number of state steps at each development step to provide a life-time evaluation. As such, the fitness function is contributed by, for the purpose of the experiments herein, counting sequence length and the sum of the longest counting sequence for each development step. The performance of the organism is based on counting sequences and the results of scaling the PDC-resources should be reflected in scaled counting behaviour.

# 5. EXPERIMENTS

The experiments presented aim to show an example of how the PDC-space can be applied to a given development model. In the approach taken the scaling regards the second challenge; scaling of the resources to achieve solution that can solve a larger problem size or a system that can solve a more complex problem stated in Section 3.4.

It is important to note that in the development model the development process and functional components (see Section 4.1) are independent. Information available from the cells neighbourhood to the development process and the functional components differ. The functional components only process information from other functional components.

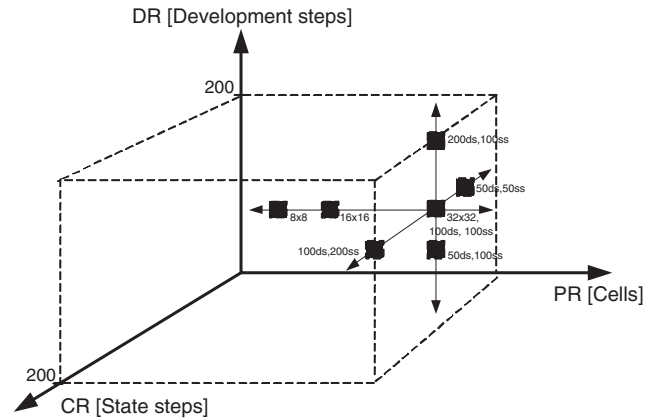
The developmental process has information from the cellular structure i.e. cell types, and state information from the functional components available. As such, the cellular neighbourhood can be scaled independently for developmental and computational resources.

## 5.1 Introducing PDC-space to the Developmental Model

The phenotypic resources of the development model presented in Section 4.1 can be given by the number of available cells in the phenotype.

Developmental resources may be scaled using several factors e.g. cellular neighbourhood or enabling/disabling environmental information. However, to investigate the scaling challenge described the development model is considered capable of generating the required structures for the functionality sought. Herein the number of developmental iterations, i.e. development steps, is chosen as the scaling factor.

The computational resources for the model may also be scaled by changing the cellular neighbourhood or by apportioned the sequential circuit phenotype more clock cycles for computation, i.e. state steps. Herein the latter is chosen.



**Figure 3: The scaling experiments placed in the PDC-space.**

Scaling of the size of the cellular array, developmental steps and state steps in the experiments can be placed in the PDC-space. In Figure 3 each experiment is shown as a black box in the PDC-space. The scaling along each axis is illustrated by the arrows.

Along the PR-axis the first experiment is given the phenotypic resources of 8 x 8 cells. The size of the cellular array is scaled to a 16 x 16 array, then after to a maximum of 32 x 32 cells. As shown the available developmental and computational resources are kept constant at a 100 DS and 100 SS respectively. Scaling of developmental resources is conducted in three scaling steps from 50 to 100 and to a maximum of 200 DS. As shown in Figure 3 this scaling experiments are conducted with the phenotypic resources set to a 32 x 32 cellular array and computational resources of 100 SS. The computational resources are likewise scaled around the 32 x 32 cellular array. The developmental resources are kept constant at 100 DS while the state steps are scaled from 50 to 100 and to a maximum of 200.

## 5.2 Experimental setup

The number of available cell types was set to thirteen including the empty cell type. Available cell types was based on Sipper's universal non-uniform CA cell [25] and threshold elements [3]. Table 1 provides the set of available cell types, together with their functional LUT definition and a graphical symbol. The first single cell which the multicellular organism develops from was defined to be a single cell of type 5 (NAND) outputting a logical "1". All empty cells are set to output a logical "0".

**Table 1: Used cell types and their functionality**

Cell type	LUT hex	Function name	Graphical representation
0	0xFFFF0000	<i>no change Emty</i>	○
1	0x66666666	$XOR_d W \oplus S$	●
2	0x3D3D3D3D	$XOR_c E \oplus S$	●
3	0x0FF00FF0	$XOR_b N \oplus E$	●
4	0x55AA55AA	$XOR_a W \oplus N$	●
5	0x55FF55FF	$NAND W \bullet N$	●
6	0xFF00FF00	↓ <i>DownPropagation</i>	●
7	0xCCCCCCCC	↑ <i>UpPropagation</i>	●
8	0xF0F0F0F0	← <i>LeftPropagation</i>	●
9	0xAAAAAAAA	→ <i>RightPropagation</i>	●
10	0xE8808000	$T \geq 4$	●
11	0xFE8E8800	$T \geq 3$	●
12	0xFFFEFE88	$T \geq 2$	●

The genome size was set to consist of 64 rules and the population size was set to 16. The initial population consisted of random generated valid rules. However, invalid rules may arise through the application of genetic operators. Crossover rate was set to 0.5 and the mutation rate for each gene was set to 0.0017. The GA was set to terminate after 100 000 generations. Each experiment consisted of ten evolutionary runs.

To be able to enhance the scaling issue a fitness function that can exploit the different scaling domains was applied. The fitness function consist of the sum of three parts; the mean accumulated longest counting sequence over all development steps, the maximum counting sequence found in the life-time of the organism and the number of active rules i.e. the number of different rules activated for the development of the organism.

The experiments were executed on a cPCI machine including a cPCI PC running the GA. Each genome was transferred on the cPCI bus to an FPGA card [23]. The development process and functional behaviour of the cellular array was executed on the FPGA [30].

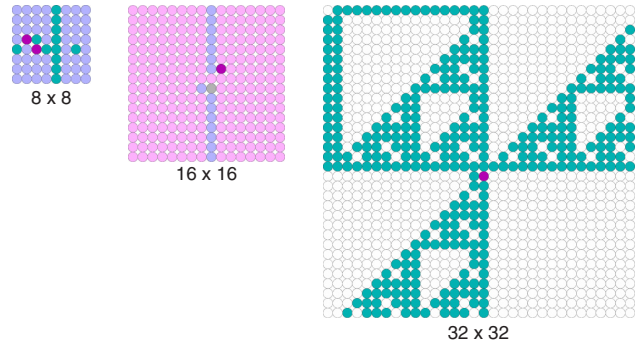
## 5.3 Phenotypic Resources

The available phenotypic resources was scaled from a minimum of 8 x 8 cells, to a phenotype of 16 x 16 cells and to a maximum phenotype with 32 x 32 cells.

Figure 4 illustrates the available size and the final phenotypes of the cellular array for the three scaling experiments regarding phenotypic resources.

In Figure 5 the results of scaling along the PR-axis are shown. The increased total fitness for the best and mean for each experiment are shown in Figure 5(a).

To further investigate the influence of scaling the mean accumulated counting sequence fitness contribution for the best result for each scale up of phenotypic resources are shown in Figure 5(b). The maximum counter sequence contribution are plotted in Figure 5(c).



**Figure 4: Final phenotypes shown for the three different scales of phenotypic resources investigated.**

The results clearly show improvements in performance for scaling up the available phenotypic resources. As shown in Figure 5(b) the mean accumulated for a 16 x 16 and 32 x 32 contribution outperform the 8 x 8 cellular array. This is as due to the longer counter sequence emerging in the larger arrays. The maximum sequence also scales with the increased phenotypic resources.

## 5.4 Developmental Resources

In Figure 6 the results of scaling along the DR-axis are shown. The comparison of the performance as a result of scaling is shown in Figure 6(a). The mean accumulated counting fitness contribution are compared in Figure 6(b). In Figure 6(c) the maximum counter sequence for each development step for the three experiments are shown.

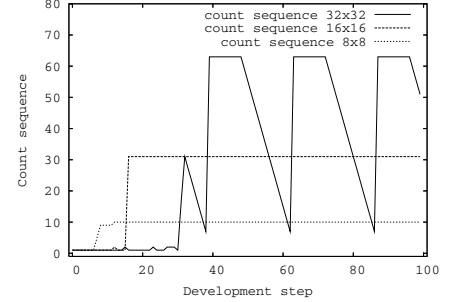
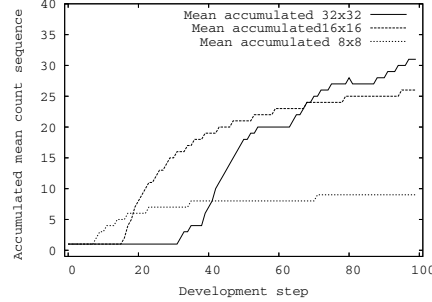
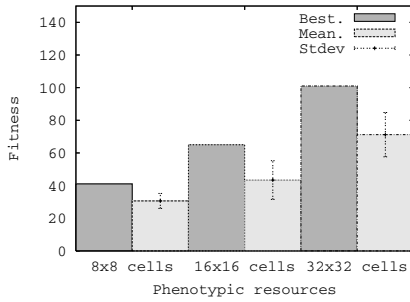
The scaling of developmental resources clearly show an improvement if the resources are scaled from 50 DS to 100 DS. However further scaling of resources to 200 DS lack improvement of performance. Figure 6(c) illustrates the lack of performance gain. The scale up of developmental resources from 100 DS to 200 DS is not exploited in the organism to create longer counting sequences. However the scale up creates long counting sequences that are stable over a greater number of development steps.

Note that the vertical lines in Figure 6(b) and 6(c) are placed to enhance the end of the organisms development (life-time) for 50 and 100 DS.

## 5.5 Computational Resources

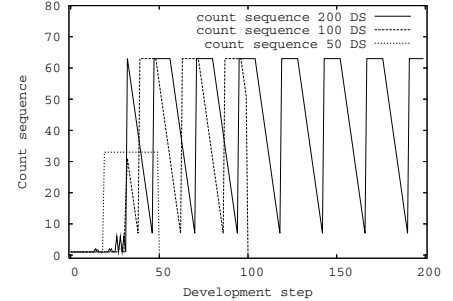
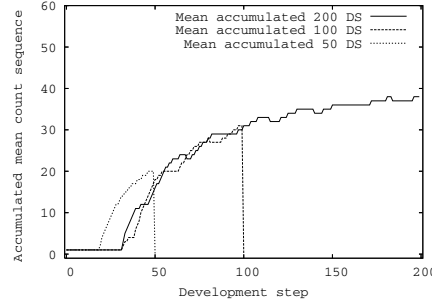
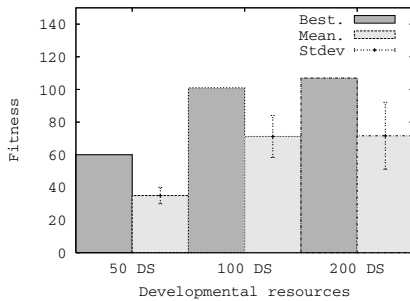
In Figure 7 the results of scaling along the CR-axis are shown. As for the previous experiments; Figure 7(a) compare the results of the scaling. Figure 7(b) plots the mean accumulated counter fitness contribution and Figure 7(c) show the longest counter found.

As for the experiment regarding development resources, the gain contribution from the last scaling step, i.e. from 100 SS to 200 SS, show little improvement in performance. However the increased computational resources were exploited to the emergence of a stabile long counter from development step 34. Note that this is only reflected in the accumulated mean sequence since the maximum counting length was equal. The increased computational resources were not exploited to create longer sequences.



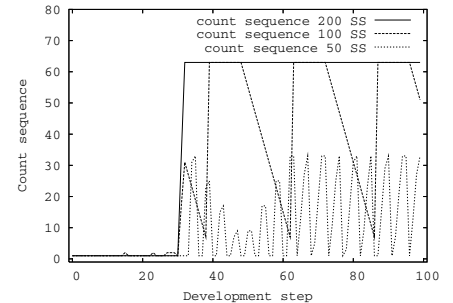
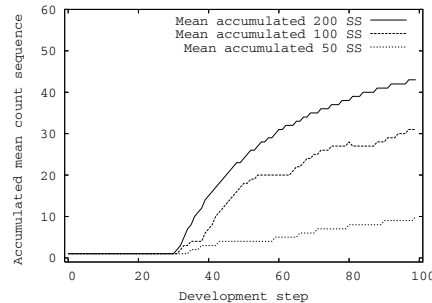
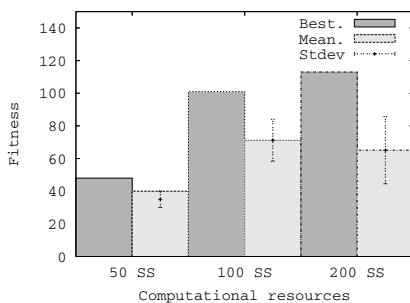
(a) Comparing the performance of the best and mean over ten runs. (b) Plot of the mean counter sequence fitness contribution. (c) Plot of the fitness contribution of the longest counter sequence found in the developing organism.

**Figure 5: Scaling the phenotypic resources from a cellular array consisting of 8 by 8 to 32 by 32 available cells.**



(a) Comparing the performance of the best and mean over ten runs. (b) Plot of the mean counter sequence fitness contribution. (c) Plot of the fitness contribution of the longest counter sequence found in the developing organism.

**Figure 6: Scaling the developmental resources from 50 to 200 development steps.**



(a) Comparing the performance of the best and mean over ten runs. (b) Plot of the mean counter sequence fitness contribution. (c) Plot of the fitness contribution of the longest counter sequence found in the developing organism.

**Figure 7: Scaling the computational resources from 50 to 200 state steps.**

## 6. CONCLUSION

This paper sought to investigate scalability in artificial developmental systems. The approach taken decomposes resources in to separate domain as to be able to allocate resources within the development model or to scale up the performance for a given problem. The three domains of scaling; phenotypic, developmental and computational resources were applied to a cellular model to demonstrate how the model can be placed in a PDC-resource space. However the approach of the proposed resource domains could be applied to most developmental models in order to scale the resources to achieve a structural and functional phenotype that can be applied to the problem at hand.

The experimental approach of placing the cellular model in the PDC-space showed how the scaling of the different resource domains can influence the result of scaling. As such, the proposed decomposition of resources can help attacking the problem of scalability in artificial development systems more systematically. Such an approach can help to characterise what resources that have most influence on the model towards solving a given problem or to expand the problem size by scaling up the performance of the system.

## 7. REFERENCES

- [1] W. R. Ashby. *An Introduction to Cybernetics*. Chapman & Hall, 1957.
- [2] J. C. Astor and A. C. A developmental model for the evolution of artificial neural networks. *Artificial Life*, 6(3):189–218, 2000.
- [3] V. Beiu, J. M. Yang, L. Quintana, and M. J. Avedillo. Vlsi implementations of threshold logic-a comprehensive survey. *IEEE Transactions on Neural Networks*, 14(5):1217–1243, September 2003.
- [4] P. J. Bentley and S. Kumar. Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In *Genetic and Evolutionary Computation Conference (GECCO '99)*, pages 35–43, 1999.
- [5] A. W. Burks. *Essays On Cellular Automata*. University of Illinois Press, 1970.
- [6] E. F. Codd. *Cellular Automata*. Association for computing machinery, Inc. Monograph series. Academic Press, New York, 1968.
- [7] P. Eggenberger Hotz. Asymmetric cell division and its integration with other developmental processes or artificial evolutionary systems. In *Artificial Life IX, European Conference on Artificial Life*. MIT press, 2004.
- [8] D. Federici. Evolving developing spiking neural networks. *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, 1:543–550 Vol.1, 2-5 Sept. 2005.
- [9] D. Federici and K. Downing. Evolution and development of a multi-cellular organism: Scalability, resilience and neutral complexification. *Artificial Life*, 12(3):381–409, 2006.
- [10] D. Gajski. *Principles of Digital Design*. Prentice-Hall, 1997.
- [11] J. Gauci and K. Stanley. Generating large-scale neural networks through discovering geometric regularities. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 997–1004, New York, NY, USA, 2007. ACM.
- [12] 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.
- [13] T. G. W. Gordon and P. J. Bentley. Towards development in evolvable hardware. In *the 2002 NASA/DOD Conference on Evolvable Hardware (EHŠ02)*, pages 241 –250, 2002.
- [14] T. G. W. Gordon and P. J. Bentley. Bias and scalability in evolutionary development. In *GECCO 2005*, pages 83 – 90. ACM Press, 2005.
- [15] T. G. W. Gordon and P. J. Bentley. Development brings scalability to hardware evolution. In *the 2005 NASA/DOD Conference on Evolvable Hardware (EHŠ05)*, pages 272 –279. IEEE, 2005.
- [16] P. C. Haddow and J. Hoyer. Achieving a simple development model for 3d shapes: are chemicals necessary? In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1013–1020, New York, NY, USA, 2007. ACM.
- [17] P. C. Haddow and G. Tufte. An evolvable hardware FPGA for adaptive hardware. In *Congress on Evolutionary Computation(CEC00)*, pages 553–560. IEEE, 2000.
- [18] S. L. Harding, J. F. Miller, and W. Banzhaf. Self-modifying cartesian genetic programming. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1021–1028, New York, NY, USA, 2007. ACM.
- [19] M. Kirschner and J. Gerhart. Evolvability. *Proceedings of the National Academy of Sciences of the United States of America*, 95(15):8420–8427, July 1998.
- [20] H. Kitano. Designing neural networks using genetic algorithms with graph generation systems. *Complex Systems*, 4(4):461–476, 1990.
- [21] H. Kitano. Building complex systems using development process: An engineering approach. In *Evolvable Systems: from Biology to Hardware, ICES*, Lecture Notes in Computer Science, pages 218–229. Springer, 1998.
- [22] J. F. Miller. Evolving a self-repairing, self-regulating, french flag organism. In *Genetic and Evolutionary Computation (GECCO 2004)*, Lecture Notes in Computer Science, pages 129–139. Springer, 2004.
- [23] Nallatech. *BenERA User Guide*, nt107-0072 (issue 3) 09-04-2002 edition, 2002.
- [24] L. Sekanina and M. Bidlo. Evolutionary design of arbitrarily large sorting networks using development. *Genetic Programming and Evolvable Machines*, 6(3):319–347, 2005.
- [25] M. Sipper. *Evolution of Parallel Cellular Machines The Cellular Programming Approach*. Springer-Verlag, 1997.
- [26] M. Sipper. The emergence of cellular computing. *Computer*, 32(7):18–26, 1999.
- [27] W. M. Spears. Gac ga archives source code collection webpage. <http://www.aic.nrl.navy.mil/galist/src/>, 1991.
- [28] L. Spector and K. Stoffel. Ontogenetic programming. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 394–399, Stanford University, CA, USA, 28–31 1996. MIT Press.
- [29] G. Tufte. Cellular development: A search for functionality. In *Congress on Evolutionary Computation(CEC2006)*, pages 2669–2676. IEEE, 2006.
- [30] 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.
- [31] G. Tufte and P. C. Haddow. Towards development on a silicon-based cellular computation machine. *Natural Computation*, 4(4):387–416, 2005.
- [32] G. Tufte and P. C. Haddow. Extending artificial development: Exploiting environmental information for the achievement of phenotypic plasticity. In *7th International Conference on Evolvable Systems (ICES07)*, Lecture Notes in Computer Science, pages 297–308. Springer, 2007.
- [33] G. Tufte and J. Thomassen. Size matters: Scaling of organism and genomes for development of emergent structures. In *CODESOAR at Genetic and Evolutionary Computation (GECCO 2006)*. ACM, 2006.
- [34] L. Wolpert. *Principles of Development, Second edition*. Oxford University Press, 2002.