Evolutionary Growth of Genomes for the Development and Replication of Multicellular Organisms with Indirect Encoding

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

***Abstract* — The genomes of biological organisms are not fixed in size. They evolved and diverged into different species acquiring new genes and thus having different lengths. In a way, biological genomes are the result of a self-assembly process where more complex phenotypes could benefit by having larger genomes in order to survive and adapt. In the artificial domain, evolutionary and developmental systems often have static size genomes, e.g. chosen beforehand by the system designer by trial and error or estimated a priori with complicated heuristics. As such, the maximum evolvable complexity is predetermined, in contrast to open-ended evolution in nature. In this paper, we argue that artificial genomes may also grow in size during evolution to produce high-dimensional solutions incrementally. We propose an evolutionary growth of genome representations for artificial cellular organisms with indirect encodings. Genomes start with a single gene and acquire new genes when necessary, thus increasing the degrees of freedom and expanding the available search-space. Cellular Automata (CA) are used as test bed for two different problems: replication and morphogenesis. The chosen CA encodings are a standard developmental table and an instruction based approach. Results show that the proposed evolutionary growth of genomes' method is able to produce compact and effective genomes, without the need of specifying the full set of regulatory configurations.**

Keywords — Artificial Development, Evolution, Replication, Complexification, Cellular Automata, Instruction-based Approach.

I. Introduction

Artificial Evolutionary and Developmental (EvoDevo) systems target principles and properties that are present in natural biological systems, such as embryogenesis, self-organization, plasticity, genotype-to-phenotype mappings, and self-repair. A major challenge for EvoDevo is the development of complex morphologies and structures, potentially at natural levels of complexity [32]. Biological multi-cellular organisms made of trillions of cells grow from a single cell which holds the complete genome [3]. However, organisms of different species have genomes of different lengths. Speciation highly relied on a gene duplication mechanism [18]. In fact, in biological organisms there is a small probability that when a parental gene is copied, more than a copy may arise, i.e. gene duplication. As such, biological evolution is an incremental process that builds up genomes of increasing complexity.

In contrast, artificial systems often rely on direct one-to-one encodings, i.e. genotype-to-phenotype mapping, where the full set of genes is mapped to the phenotype entities directly. This solution has several drawbacks, for instance scaling up the phenotype resources would imply a larger genotype which would result in a larger search space. Another possibility is to open for an indirect encoding, i.e. development or generative mapping. Such mappings are a necessity when then number of phonotype entities is large [33]. However, the scalability of such systems is still an open challenge. In fact, for several developmental systems [20, 37], it would not be possible to represent all the possible regulatory combinations in the genotype. The system designer would need to make assumption or use complex heuristic to find out a reasonable genome size, such that it would be large enough to contain a potential solution [5, 37, 38].

In this paper, we propose a biologically inspired evolutionary growth of genomes with indirect encoding, which does not rely on any a priori knowledge on the problem complexity or required genome size. Genotypes are initialized with a single gene and genotype size is incrementally evolved by means of gene duplications. This would guarantee a low dimensionality of the search space that would, in turn, evolve increasingly complex solutions and add newly created degrees of freedom only when needed. The proposed framework is tested on two different problems: the development of given patterns starting from an initial seed [9] and the replication of cellular structures [15]. Self-replication allows reuse of genetic material, reducing genome size and enabling the production of complex structures.

Related work was done towards achieving genome size expansion [6, 12, 14, 17] with variable length genomes, mostly using direct encodings [10, 34]. Research on complexification with direct encodings may not be conclusive but the idea of incremental evolutionary growth of genomes has a potential for exploration with indirect encodings [32]. With direct encodings, a variable length genome allows to duplicate a gene that is mapped to a single phenotypic entity. This would result in a duplication of the correspondent phenotypic structure itself. Such approach has been shown to be particularly successful when modular structures ought to be evolved [7]. In contrast, with indirect encodings, adding a new gene may have

disruptive effects, since genes have shared and overlapping regulations. A challenge is to allow the newly added gene enough time to be incorporated and optimized in the genome. An important property that allows such optimization is neutrality [29, 30].

The paper is laid out as follows: section 2 provides background information on genome expansion and instruction-based development. Section 3 presents the cellular developmental model and the framework for evolutionary growth of genomes. In section 4 the problems under investigation are described and section 5 presents the experimental setup. Section 6 provides the experimental results together with discussion and Section 7 concludes the work.

II. Related Work

A. Growing Genome

The mechanism of genome expansion through gene duplication played a key role in natural evolution [35, 28]. There is strong evidence that around 38% of the Homo Sapiens genome is a result of gene duplication [16]. Such process contributed to the emergence of complex regulatory mechanisms in gene regulatory networks, providing plasticity and adaptivity. In the artificial domain, previous work was done towards using variable length genomes [36] and achieving genome size expansion [6, 12, 14, 17, 1, 2]. Federici and Downing [10] investigated neutral gene duplications in a direct encoding cellular model with environmental chemicals. Stanley and Miikkulainen [34] introduced NeuroEvolution of Augmenting Topologies (NEAT), a complexification method for the incremental evolution of neural network topologies. Their main goal was to evolve robot controllers with direct encodings through gene duplications. A CA framework for the evolutionary growth of genomes has been implemented with indirect encodings by Nichele and Tufte [21]. The goal was to evolve trajectories and attractors of different lengths. This approach has been shown to be suitable to evolve cellular automata local transition functions starting from a single neighborhood configuration, being able to scale well when the search space, the state space or the phenotype resources were scaled up. As a step forward, we investigate the proposed framework in [21] with an advanced genotype-to-phenotype indirect encoding, namely the instruction-based approach.

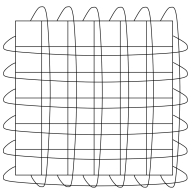
B. Instruction-based Approach

The idea of evolving instruction-based representations is a rather old approach [13]. Cartesian Genetic Programming (CGP) has been introduced by Miller and Thomson [19] for the evolution of digital circuits by representing a program as a directed graph. Sipper [31] proposed the evolution of non-uniform cellular automata with cellular programming, an evolutionary approach based on CA propagation, NAND and XOR operations. This has been proven to be computation universal. Bidlo and Skarvada [4] introduced the Instruction-Based Development (IBD) for the evolutionary design of digital circuits. Bidlo and Vasicek [5] exploited IBD for the development and replication of cellular automata structures. Even though their approach has been shown to improve the overall success rate, the number of available instruction was arbitrarily chosen. IBD allows representing CA transition functions by means of a sequence of instructions, i.e. a program, which is executed on local neighborhoods in parallel and deterministically determines the next state of each cell. In contrast, a traditional CA local transition function specifies all the possible neighborhood combinations together with the next state of the cell being considered. If the number of cell states of neighborhood size is increased, specifying all the neighborhoods may become not feasible. Evolutionary growth of genomes has been used to evolve local transition functions starting from a single neighborhood configuration [21]. Using an instruction-based approach removes the problem of specifying all the combinations in the transition function, but the problem of determining the number of necessary instructions to solve a given problem still exists. As such, we propose an IBD based on evolutionary growth of genome.

III. Cellular Developmental Model

The cellular developmental model is based on 2-dimensional cellular automata, with synchronized cellular cycle, parallel updates and discrete cell states. For more details on the model see [21, 24, 23, 22, 25]. A CA can be considered as a developing organism, where the genome specifications and gene regulation information control the cell's growth and differentiation. The behavior of the CA is represented by the emerging phenotype, which is subject to size and shape modifications. The cellular model has cyclic boundary conditions and uses von Neumann neighborhood (5 neighbors, i.e. up, down, left, right and centre cell).

A standard table-based CA transition function would consist of the specification of all the possible neighborhood configurations, together with the state of the central cell at the next time-step. This is shown in Figure 1, for a CA with n cell states. The cell type *0* is defined as the empty state, i.e. quiescent. The column *C(t+1)* can be any of the possible cell states. As such, the table-based representation would consist of n^5 possible neighborhood configurations (with n possible cell states and *5* neighbors). If a CA transition table ought to be evolved, with 4 cell states and 5 neighbors, the total number of different transition functions, i.e. search space, would be $4^{4^5} = 4^{1024} = \sim 3.23 \times 10^{616}$.



L	R	U	D	C	C(t+1)
0	0	0	0	0	0
0	0	0	0	1	{0,1,2, ..., n}
0	0	0	1	0	{0,1,2, ..., n}
0	0	0	1	1	{0,1,2, ..., n}
0	0	1	0	0	{0,1,2, ..., n}
:	:	:	:	:	:
1	1	1	1	1	{0,1,2, ..., n}
0	0	0	0	2	{0,1,2, ..., n}
0	0	0	2	0	{0,1,2, ..., n}
0	0	0	2	1	{0,1,2, ..., n}
0	0	0	2	2	{0,1,2, ..., n}
:	:	:	:	:	:
n-1	n-1	n-1	n-1	n-1	{0,1,2, ..., n}
0	0	0	0	n	{0,1,2, ..., n}
0	0	0	n	0	{0,1,2, ..., n}
0	0	0	n	1	{0,1,2, ..., n}
0	0	0	n	2	{0,1,2, ..., n}
:	:	:	:	:	:
n	n	n	n	n	{0,1,2, ..., n}

Fig. 1. CA model with cyclic boundaries (left), von Neumann neighborhood (centre), table-based transition function for the CA development.

For the instruction-based approach, each gene in the genotype would consist of an instruction with a maximum of

two operands chosen among the neighbors, as specified in Table 1. In our model, there are a total of 16 instructions and each operand can be one of the five neighbors. As such, the search space for each instruction is 16x5x5= 400. Evolutionary growth of genomes is initialized by a single instruction and grows the program size incrementally. A developmental genome composed by 5 instructions (as some results in this paper) would have a search space of $400^5 = 1.024 \times 10^{13}$ (many orders of magnitude less than the traditional method).

TABLE I. THE INSTRUCTION SET FOR THE CELLULAR DEVELOPMENT PROCESS. $N(op_1)$ AND $N(op_2)$ REPRESENT TWO CELL STATES IN THE NEIGHBORHOOD FOR THE TWO OPERANDS OF THE INSTRUCTION. n REPRESENTS THE NUMBER OF CELL STATES. ALL OPERATIONS ARE PERFORMED MODULO(n) TO GUARANTEE A PERSITENT RESULT.

Instruction	Description	Meaning	Code
AND	$N(op_1) = N(op_1) \wedge N(op_2)$	AND operation	0
OR	$N(op_1) = N(op_1) \vee N(op_2)$	OR operation	1
XOR	$N(op_1) = N(op_1) \oplus N(op_2)$	XOR operation	2
NOT	$N(op_1) = \neg N(op_1)$	NOT operation	3
INV	$N(op_1) = n - N(op_1)$	Inverse state	4
MIN	$N(op_1) = \min(N(op_1), N(op_2))$	Minimum	5
MAX	$N(op_1) = \max(N(op_1), N(op_2))$	Maximum	6
SET	$N(op_1) = N(op_2)$	Set value	7
INC	$N(op_1) = N(op_1) + 1$	Increment	8
DEC	$N(op_1) = N(op_1) - 1$	Decrement	9
SWAP	$N(op_1) \leftrightarrow N(op_2)$	Swap	10
ROR	LCR → RLC	Rotate right	11
ROL	LCR → LCR	Rotate left	12
ROU	UCD → CDU	Rotate up	13
ROD	UCD → DUC	Rotate down	14
NOP	$N(op_1) = N(op_1)$	No operation	15

The resulting program is executed in parallel on a local neighborhood copy for each CA cell. For example, if the program (MIN UP LEFT) (XOR CENTRE UP) is executed, a local copy for each CA cell's neighborhood is considered (5 neighbors): UP, RIGHT, DOWN, LEFT, and CENTRE (the central cell being considered). The first executed instruction is UP = MIN(UP, LEFT) which updates the UP neighbor. The second is CENTRE = XOR(CENTRE, UP), which updates the CENTRE neighbor. At the end of the program, the state of the CENTRE cell is copied back as the next state (the result of the transition function).

A. *Evolutionary growth of genomes*

A conventional Genetic Algorithm (GA) is used for the evolution of table-based transition functions. For the instruction-based transition function the genotype information is composed by a sequence of genes that represent a developmental program (each gene codes for an instruction with relative operands). As such, the GA is modified with the introduction of the following regulation mechanisms:

1. There is an upper bound on the number of genes that can be added, which coincides with the maximum possible neighborhoods for the table-based transition function;
2. There is a duplication rate which guarantees that a new gene is added with a certain probability to each individual in the population;
3. Added genes are guaranteed time for optimization, before a new addition can occur. This means that there is a counter and a threshold which specifies that a genome growth can occur only after a certain number of generations without an increase in the overall best fitness in the population (optimization time threshold). Duplication happens when there is no room for further genotype optimization;
4. There is an elitism mechanism. This is done to guarantee genotypes with compact and effective genomes will be kept in the population no matter what. This enables speciation and survival of fit individuals with smaller genomes.

If the four regulation mechanisms hold, a gene addition occurs. A genotype that consists of n genes at a given generation acquires a new gene which is an exact copy of a randomly selected gene already present in the genome. The selected gene is copied and appended; the genotype will then consist of n+1 genes.

In the process of selecting individuals, either for offspring generation or for gene addition, a weighted selection is implemented with the actual fitness that counts 80% and the innovation parameter that rewards larger genomes, which counts 20%. The innovation parameter allows newly added genes, which most likely result in a fitness-neutral or fitness-decrease, to be allowed to survive and be optimized. For more details on the framework for evolutionary growth of genomes, please refer to [21].

IV. PROBLEMS UNDER INVESTIGATION

A traditional table-based CA evolution is compared to an evolutionary growth of genomes in terms of genotypes size (number of necessary genes), success rate and number of generations. The following two problems were investigated:

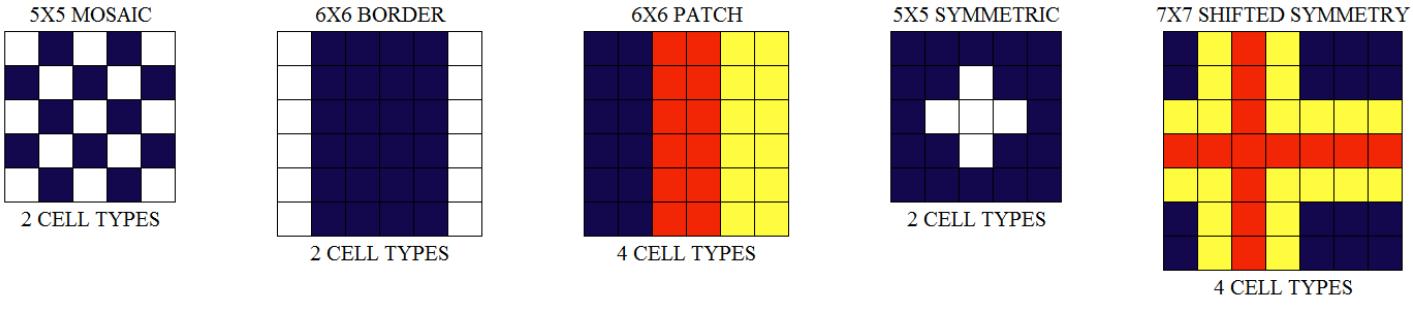


Fig. 2. Target structures for the development and replication problems.

1. Morphogenesis: a target structure develops out of a seed (of non-quiescent type) placed in the centre of the grid, while all the other cells are in the quiescent state. There is a maximum number of development steps in which the target pattern may develop. The size of the grid is the same as the size of the target structure.
2. Replication: a structure placed in the centre of the grid is to be replicated in a given number of development steps. At least 3 replicas of the initial structure are required;

The target structures are represented in Figure 2 and have been chosen according to widely used examples in literature [36, 8, 20, 11, 38]. In particular, they are chosen to cover a variety of properties, complexities and characteristics of developmental systems, such as self-organization, modularization, and diversification. The identified classes are: 2a) mosaic pattern (5x5 with 2 cell types), 2b) border pattern (6x6 with 2 cell types), 2c) patch pattern (6x6 with 2 cell types), 2d) point-symmetric pattern (5x5 with 2 cell types), 2e) shifted symmetry pattern (7x7 with 4 cell types).

Figure 3a) represents the initial seed for the morphogenesis problem. In the target structures to be replicated or developed, the available number of cell types is set to 2 or 4, as represented in Figure 3b), which are mapped to the cell types in Figure 2.

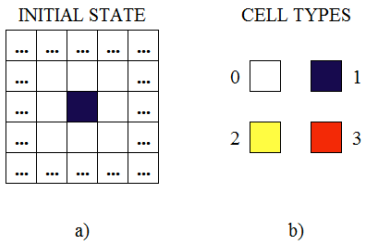


Fig. 3. a) initial state for the morphogenesis problem. b) available cell types

For the morphogenesis problem, candidate solutions are evaluated with the maximum value of all the partial matches in all the development steps. For the replication problem, the whole grid is inspected to find perfect replicas of the target structure in each development step and the 3 top matches (perfect or partial) are considered. A bonus is given if a perfect replica is found. The used fitness function for the replication and development problem are similar to those described in [5].

V. Experimental Setup

In both test cases (morphogenesis and replication), transition functions are evolved to solve the target problem (develop or replicate one of the structures in Figure 2). All the experiments are executed for 100 independent GA runs. Table-based rules and program-based rules are evolved for a maximum of 10000 generations, with standard GA and evolutionary growth of genomes, respectively. If a solution has not been found in 10000 generations, the experiment is considered not successful. The population of genotypes, i.e. transition functions, consists of 30 individuals for the development problem and 16 individual in the replication problem. Each CA genotype rule is executed and evaluated for 30 development steps. The GA selection process uses proportionate selection base 4 (four individuals are picked randomly and the best among them is selected). In addition, one elite is always transferred to the offspring. For the standard table-based transition function, all the possible neighborhood configurations are explicitly represented, together with the correspondent next value of the cell under consideration. The chosen mutation rate for each of the next cells' states is 0,02 for the structures with 2 cell types and 0,066 for the structures with 4 cell types (see Figure 2). For the program-based transition function, the genotype size grows along evolution. Each of the genes represents an instruction, i.e. an instruction code and two operands. As such, 2 mutations each genotype, i.e. program, are performed. Both operation codes and operands can be selected for mutation, within the correct range. Crossover is not suitable for our experiments (due to variable length genomes) and thus it is not included.

A. *Morphogenesis*

For the morphogenesis problem, the target patterns are the following (from Figure 2):

- 2a) mosaic pattern of size 5x5 with 2 cell types;
- 2b) border pattern of size 6x6 with 2 cell types;
- 2c) patch pattern of size 6x6 and 4 cell types (well known as the French Flag problem [27]);
- 2d) symmetric pattern of size 5x5 and 2 cell types.

In all the cases, the initial CA state is a zygote cell of type 1, as represented in Figure 3a and all the other cells in the grid are of type 0 (quiescent). The CA size is set to the target pattern size. Genomes for the table-based transition function are initialized by listing all the possible neighborhood configurations and the next value of the considered cell randomly generated among the available cell types. As such, the total number of genes is $2^5 = 32$ (with 2 cell types and 5 neighbors) and $4^5 = 1024$ (with 4 cell types and 5 neighbors).

For the program-based transition function, the genomes are initialized with a single randomized gene (random instruction code and operands). Genomes grow during evolution using the described evolutionary growth of genomes framework with the following regulation parameters:

1. Genes number upper bound, i.e. maximum program length, of 32 and 1024 instructions for 2 and 4 cell types' structures respectively. This coincides with the total number of neighborhood combinations for a classical table-based CA;
2. Duplication rate, i.e. rate at which a gene can be added, of 1/(population size). On average, one out of the 30 genomes acquires a new gene, if the following point is true;
3. Optimization time threshold, i.e. number of generations without overall fitness increase, of 30 generations for structures 2a) and 2d), 100 generations for structures 2b) and 2c);
4. Elitism of 1 individual in the population which guarantees that the fittest survives no matter what.

B. Replication

For the replication problem, the structures to be replicated are (from Figure 2):

- 2a) mosaic pattern of size 5x5 with 2 cell types, in a grid of size 27x27 cells (enough space to replicate on all sides);
- 2c) patch pattern of size 6x6 and 4 cell types (well known as the French Flag problem [27]), in a grid of size 30x30 cells;
- 2d) symmetric pattern of size 5x5 and 2 cell types, in a grid of size 27x27 cells;
- 2e) shifted symmetry pattern of size 7x7 and 4 cell types, in a grid of size 33x33 cells.

The minimum number of wanted replicas is set to 3 and each replica has to have a border of quiescent cells (type 0) that surrounds the structure. The initialization of genomes is the same as for the morphogenesis problem and the evolutionary growth of genomes has the following parameters:

1. Genes number upper bound of 32 and 1024 instructions for 2 and 4 cell types' structures respectively;
2. Duplication rate of 1/(population size = 16);
3. Optimization time threshold of 10 generations;
4. Elitism of 1 individual.

VI. Results

A. Morphogenesis

The results for the morphogenesis problem are summarized in Table II, where the traditional table-based evolution is compared with the instruction-based growing evolution for each of the target structures to be developed.

As expected, the growing evolution function has a much higher success rate.

Looking at the evolved genotype size, the table-based evolution has fixed number of genes, defined by the possible neighborhood configurations. The growing evolution function is able to achieve optimized genomes which are compact and effective. In particular, the patch structure 2c) would need 1024 genes for the table-based evolution whether it need only 19.65 genes on average for the growing evolution function. The differences in genotype size between table-based evolution and instruction-based growing evolution are significant (Student t-test, $p<0.0001$) in all the analyzed cases. Moreover, the necessary number of generations that evolution needs to find solutions is not larger for the growing evolution function. The difference in the average number of generations is not statistically significant (Student t-test). Figure 4 shows a bar plot where the differences in genotype size are summarized for the morphogenesis problem. In conclusion, it is possible to notice that for structures with 2 cell types, the average number of necessary genes is halved. For structure 2c) which has 4 cell types the shrinking factor is more accentuated (y-axes has log. scale). An example of evolved solution is shown in Figure 8.

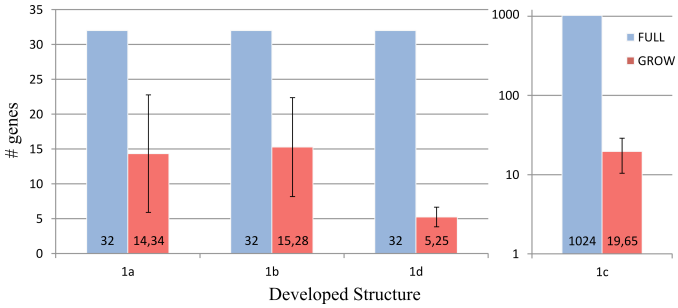


Fig. 4. Average genotype size for the morphogenesis problem. Table-based evolution in blue and growing evolution in red.

B. Replication

Table III summarizes the results for the replication problem. Again here the growing evolution function has higher success rate, being able to achieve solutions for all structures with a success rate of 100%.

TABLE II. Morphogenesis (avg. 100 runs)

Fig.	Table-based Evolution						
	Success Rate %	*Genotype Size (# genes)*				*Generations*	
		Max	*Avg*	*Min*	*StDev*	*Avg.*	*StDev.*
2a	58	32	32	32	0	1336	2294
2b	69	32	32	32	0	2254	2501
2c	19	1024	1024	1024	0	5002	3157
2d	23	32	32	32	0	2668	2942
Fig.	**Instruction-based Growing Evolution**						
	Success Rate %	*Genotype Size (# genes)*				*Generations*	
		Max	*Avg*	*Min*	*StDev*	*Avg.*	*StDev.*
2a	98	31	14.34	5	8.4318	1257	1152
2b	98	31	15.28	5	7.0973	3956	1690
2c	46	46	19.65	6	9.2236	6424	1922
2d	100	13	5.25	4	1.4097	285	108

TABLE III. Replication (avg. 100 runs)

Fig.	Table-based Evolution						
	Success Rate %	*Genotype Size (# genes)*				*Generations*	
		Max	*Avg*	*Min*	*StDev*	*Avg.*	*StDev.*
2a	85	32	32	32	0	775	1393
2c	8	1024	1024	1024	0	4331	3576
2d	1	32	32	32	0	8259	0
2e	0	1024	1024	1024	0	-	-
Fig.	**Instruction-based Growing Evolution**						
	Success Rate %	*Genotype Size (# genes)*				*Generations*	
		Max	*Avg*	*Min*	*StDev*	*Avg.*	*StDev.*
2a	100	7	2.93	2	1.1742	39.7	19.6
2c	100	6	2.84	2	1.1166	39.6	22.3
2d	100	8	3.06	2	1.2128	41.8	20.5
2e	100	5	1.38	1	0.8012	9.4	10.7

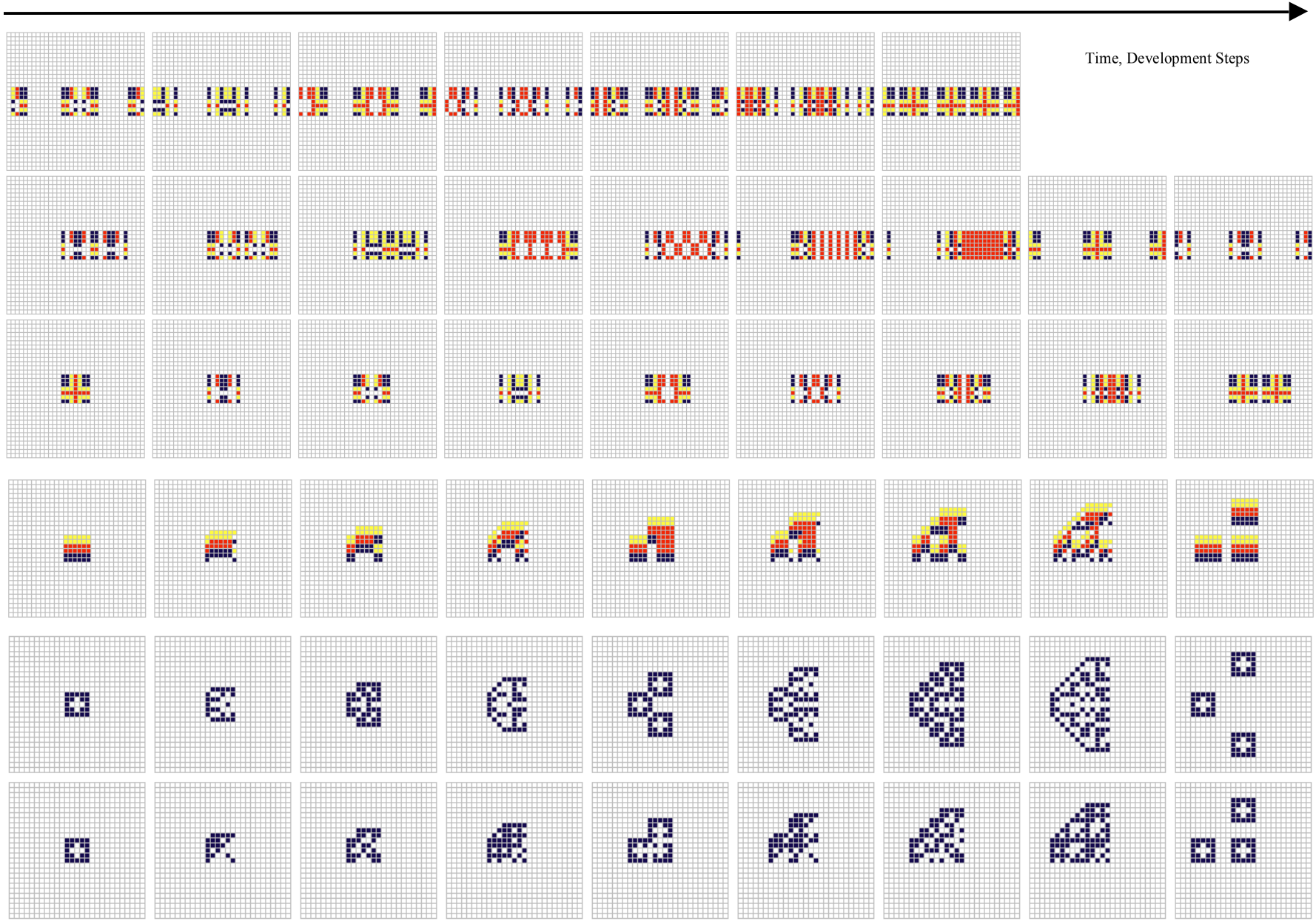


Fig. 5. Examples of evolved solutions for the replication of the structures 2d, 2c and 2e. Three perfect replicas are required and each of them is a replicator itself.

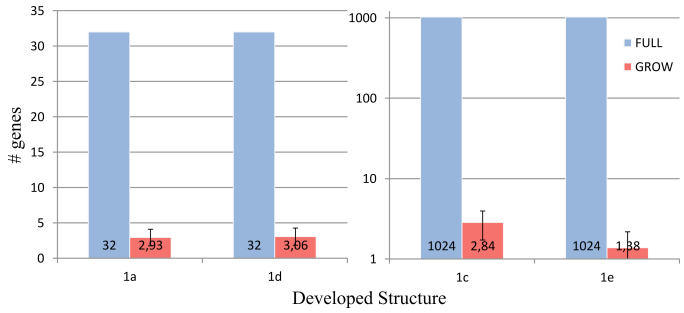


Fig. 6. Average genotype size for the replication problem. Table-based evolution in blue and growing evolution in red.

With table based-evolution, the only structure with a high success rate (85%) is 2a). For structure 2c), solutions were found only 8% of times and for structure 2d) one a single solution out of 100 runs. For structure 2e) no solution was found by the table-based evolution. Analyzing the average genotype size for both methods, it is evident that growing evolution is able to evolve particularly optimized genomes, being able to evolve in most cases the xor-based (addition followed by modulo division) replicator. As such, having a search space that grows incrementally makes it possible to add degrees of freedom during evolution only when needed. This can be observed by the number of needed generations to find solutions, which is much lower for the growing evolution function, which performs better both in terms of overall genome size and needed generations. The differences are statistically significant (Student t-test, $p<0.0001$) in all cases (t-test not calculated when less than 2 solutions were available, with structure 2d) and 2e) for the table-based evolution).

Figure 6 shows a bar graph comparison of the evolved genome size for a full table (table-based evolution function) and growing evolution, where it is possible to notice graphically the big difference in number of used genes. Four examples of evolved solutions for the replication of structures 2d, 2c and 2e are shown if Figure 5. The first two examples show two different strategies for the replication of the symmetric structure in Figure 2d. In both cases, the result after 9 development steps is 3 perfect copies of the initial structure. The third example shows the replication of the patch structure in Figure 2d (also known as the French Flag problem). The last example plots the replication of the shifted symmetric structure in Figure 2e, where 25 development steps are necessary to obtain 3 perfect replicas. The replication process does not show any repetitive instruction sequence or construction arm. This is consistent with [26] as "…the structures replicated via a fission process in which highly parallel processing occurs…an initial structure would typically grow and then divide, making replication very fast".

Figure 7 plots the evolutionary growth of genome size together with the increasing normalized fitness for the replication of the patch pattern in Figure 2c). Results are consistent with [21], where the genomes' build-up process happens in the first phase of evolution and then it shows a tendency to optimize and add fewer genes in the second half. This self-organizing process is the result of the implemented regulation mechanisms for the evolutionary growth of genomes (see Section III-A).

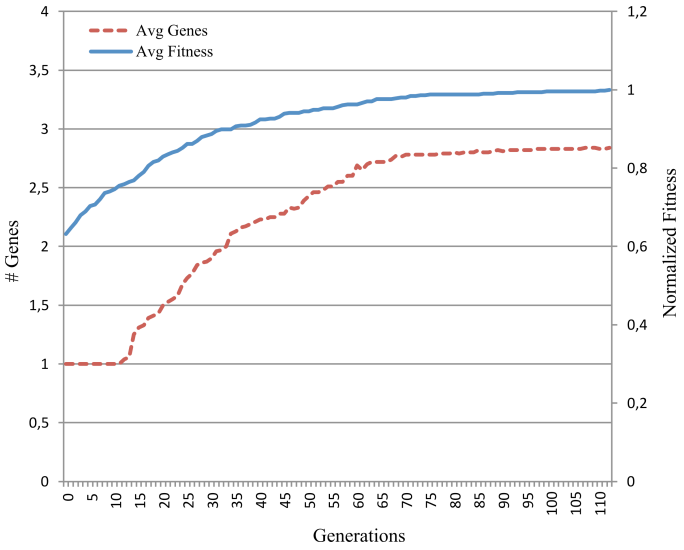


Fig. 7. Evolved genome size and normalized fitness for the replication of structure 2c – patch structure (avg. over 100 runs)

VII. Conclusion

In this paper we presented an evolutionary growth of genome size where the initial genotype is initialized with a single random gene. New genes are added by means of gene duplication when the available genes have been optimized. This incremental process of genome expansion is inspired by biological evolution where different species evolved genomes of different lengths. The proposed evolution is implemented with an indirect encoding that exploits an instruction-based development. The obtained genomes have been shown to be more compact and effective compared to a standard CA table evolution, where all the regulatory combinations are fully specified. The proposed method showed better success rate on average for the development problem (development of a given structure starting from a zygote) and for the replication problem (produce at least three exact copies of a given structure). As future work, it may be interesting to optimize and reduce the instruction set. This was out of the scope of the experiments herein. Another interesting approach would be to test the presented method on other computational tasks such as the rotation or mirroring of structures.

Evolutionary growth of genomes has been shown to be a powerful complexification tool for cellular systems with indirect encodings. It may be interesting to let the growth happen also in the number of available cell types, i.e. no boundaries on the state space. Another future direction is to introduce instructions that can modify the program itself, as in Self-Modifying Cartesian Genetic Programming (SMCGP). This would allow the diversification of cells' programs.

References

[1] L. Altenberg. Evolving better representations through selective genome growth. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 182–187. IEEE, 1994.

[2] L. Altenberg. Genome growth and the evolution of the genotype-phenotype map. In W. Banzhaf and F. H. Eeckman, editors, *Evolution and Biocomputation*, volume 899 of *Lecture Notes in Computer Science*, pages 205–259. Springer, 1995.

[3] E. Bianconi, A. Piovesan, F. Facchin, A. Beraudi, R. Casadei, F. Frabetti, L. Vitale, M. C. Pelleri, S. Tassani, F. Piva, S. Perez-Amodio, P. Strippoli, and S. Canaider. An estimation of the number of cells in the human body. *Annals of Human Biology*, 0(0):1–9, 2013. PMID: 23829164.

[4] M. Bidlo and J. Skarvada. Instruction-based development: From evolution to generic structures of digital circuits. *KES Journal*, 12(3):221–236, 2008.

[5] M. Bidlo and Z. Vasicek. Evolution of cellular automata using instruction-based approach. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8, 2012.

[6] D. Cliff, P. Husbands, and I. Harvey. Explorations in evolutionary robotics. *Adaptive Behaviour*, 2(1):73–110, 1993.

[7] J. Clune, J.-B. Mouret, and H. Lipson. The evolutionary origins of modularity. *Proceedings of the Royal Society B: Biological Sciences*, 280(1755), 2013.

[8] A. Devert, N. Bredeche, and M. Schoenauer. Robustness and the halting problem for multicellular artificial ontogeny. *IEEE Trans. Evolutionary Computation*, 15(3):387–404, 2011.

[9] R. Doursat, H. Sayama, and O. Michel. *Morphogenetic Engineering: Toward Programmable Complex Systems*. Springer Publishing Company, Incorporated, 2013.

[10] D. Federici and K. Downing. Evolution and development of a multicellular organism: Scalability, resilience, and neutral complexification. *Artificial Life*, 12:2006, 2006.

[11] P. C. Haddow, G. Tufte, and P. van Remortel. Shrinking the genotype: L-systems for ehw? In Y. Liu, K. Tanaka, M. Iwata, T. Higuchi, and M. Yasunaga, editors, *ICES*, volume 2210 of *Lecture Notes in Computer Science*, pages 128–139. Springer, 2001.

[12] I. Harvey, P. Husbands, and D. Cliff. Seeing the light: Artificial evolution, real vision, 1994.

[13] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[14] J. R. Koza. Gene duplication to enable genetic programming to concurrently evolve both the architecture and work-performing steps . . . In *IN IJCAI-95 PROCEEDINGS OF THE FOURTEENTH INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE*, pages 734–740. Morgan Kaufmann, 1995.

[15] C. G. Langton. Self-reproduction in cellular automata. *Physica D*, 10D(1-2):135–44, 1984.

[16] W.-H. Li, Z. Gu, H. Wang, and A. Nukrutenko. Evolutionary analyses of the human genome. *Nature*, (409):847–849, 2001.

[17] K. Lindgren and J. Johansson. Coevolution of strategies in n-person prisoner's dilemma. *Evolutionary Dynamics-Exploring the Interplay of Selection*, Jan. 2001.

[18] A. P. Martin. Increasing genomic complexity by gene duplication and the origin of vertebrates. *The American Naturalist*, 154(2):pp. 111–128, 1999.

[19] J. Miller and P. Thomson. Cartesian genetic programming. In R. Poli, W. Banzhaf, W. Langdon, J. Miller, P. Nordin, and T. Fogarty, editors, *Genetic Programming*, volume 1802 of *Lecture Notes in Computer Science*, pages 121–132. Springer Berlin Heidelberg, 2000.

[20] J. F. Miller and W. Banzhaf. Evolving the program for a cell: from french flags to boolean circuits. In S. Kumar and P. J. Bentley, editors, *On Growth, Form and Computers*. Academic Press, October 2003.

[21] S. Nichele, A. Giskeødegård, and G. Tufte. Evolutionary Growth of Genome Representations on Artificial Cellular Organisms with Indirect Encodings. *SUBMITTED to Artificial Life, MIT Press*, ??, 2014.

[22] S. Nichele and G. Tufte. Genome parameters as information to forecast emergent developmental behaviors. In *Unconventional Computation and Natural Computation*, pages 186–197. Springer, 2012.

[23] S. Nichele and G. Tufte. Evolution of incremental complex behavior on cellular machines. In *Advances in Artificial Life, ECAL*, volume 12, pages 63–70, 2013.

[24] S. Nichele and G. Tufte. Measuring phenotypic structural complexity of artificial cellular organisms. In *Innovations in Bio-inspired Computing and Applications*, pages 23–35. Springer, 2014.

[25] S. Nichele, H. Wold, and G. Tufte. Investigation of genome parameters and sub-transitions to guide evolution of artificial cellular organims. In *Proceedings of the 16th European Conference on the Applications of Evolutionary Computation (EvoApplications 2014)*, 2014.

[26] Z. Pan and J. A. Reggia. Computational discovery of instructionless self-replicating structures in cellular automata. *Artificial Life*, 16:39–63, 2010.

[27] O. U. Press. *Principles of Development*. Wolpert, L., 1998.

[28] O. S. *Evolution by Gene Duplication*. Allen and Unwin, London, UK, 1970.

[29] R. Shipman. Genetic redundancy: desirable or problematic for evolutionary adaptation? In *Artificial Neural Nets and Genetic Algorithms, 1999. Proceedings of the International Conference on*, pages 337–344. Springer-Verlag, 1999.

[30] R. Shipman, M. Shackleton, M. Ebner, and R. Watson. Neutral search spaces for artificial evolution: a lesson from life. In *Artificial Life VII, Proceedings of the Seventh International Conference on Artificial Life*, pages 162–169. MIT Press, 2000.

[31] M. Sipper. *Evolution of Parallel Cellular Machines, The Cellular Programming Approach*, volume 1194 of *Lecture Notes in Computer Science*. Springer, 1997.

[32] K. O. Stanley and R. Miikkulainen. Achieving high-level functionality through evolutionary complexification. In *Proceedings of the AAAI-2003 Spring Symposium on Computational Synthesis*, Stanford, CA, 2003. AAAI Press.

[33] K. O. Stanley and R. Miikkulainen. A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130, 2003.

[34] K. O. Stanley and R. Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21:63–100, 2004.

[35] J. S. Taylor and J. Raes. Duplication and divergence: The evolution of new genes and old ideas. *Annual Review of Genetics*, 38(1):615–643, 2004. PMID: 15568988.

[36] M. A. Trefzer, T. Kuyucu, J. F. Miller, and A. M. Tyrrell. On the Advantages of Variable Length GRNs for the Evolution of Multicellular Developmental Systems. *Evolutionary Computation, IEEE Transactions on*, 17(1):100–121, Feb. 2013.

[37] G. Tufte. Discovery and investigation of inherent scalability in developmental genomes. In G. Hornby, L. Sekanina, and P. C. Haddow, editors, *ICES*, volume 5216 of *Lecture Notes in Computer Science*, pages 189–200. Springer, 2008.

[38] G. Tufte and J. Thomassen. Size matters: Scaling of organism and genomes for development of emergent structures. In *Genetic and Evolutionary Computation (GECCO)*, ACM conference series. ACM, 2006.

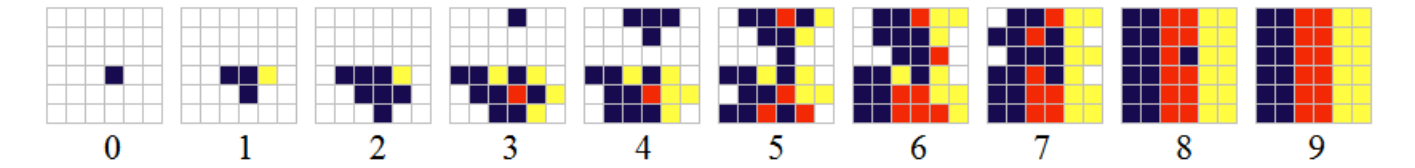


Fig. 8. Example of evolved program for the development of structure 2c – patch structure. After development step 9 the structure remains stable (point attractor). The program is composed by 14 instructions (one instruction each gene) in the following standard: INSTRUCTION CODE, OPERAND 1, OPERAND 2 (if the operand is not applicable for the given instruction, the value is ignored). For instruction codes and relative meaning see Table 1, the operands are defined as follows: UP = 0, RIGHT = 1, DOWN = 2, LEFT = 3, CENTRE = 4.