Emergent Robustness and Self-Repair through Developmental Cellular Systems

Can Öztürkeri and Mathieu S. Capcarrere
Kent University
Natural Computation Group, Computing Laboratory
Canterbury, Kent CT2 7NF, United-Kingdom
{co24, M.Capcarrere}@kent.ac.uk
http://www.cs.kent.ac.uk/research/groups/aii/ncg/

Abstract

Fault-tolerance and, even more, self-repair remain elusive properties in computing systems. In contrast, natural systems are often cited as examples of flexible, self-repairable systems. Such capabilities rely on many different aspects, but our hypothesis is that (adaptive) growth and cellularity are at the heart of these properties lacking so dearly in human artifacts. In this paper, we propose a simple cellular developmental system to back-up through experimental results this hypothesis. First, we show that it is possible to evolve such systems to do specific tasks. Second, and more importantly, that these systems exhibit *emergent* robustness and self-repair capabilities through their own nature rather than specific design or directed evolution.

Introduction & Previous Research

Fault-tolerance and, even more, self-repair remain an elusive property in computing systems. If redundancy is a necessity in a physical system, in abstract or software systems, first detection of errors and then reconstruction are the two main issues in designing robust and self-repairable systems. This reconstruction process is all the easier if the system is able to construct itself in the first place. It can be hypothesised thus, that if the system stabilises its self-construction process into a useful, working state, then any perturbations, any errors, would create an unstable state that should redevelop, restabilise, reconstruct into the desired stable working state. This approach, though still rather undeveloped, has been hinted at to a greater or lesser extent in a certain number of previous studies, (De Garis, 1999; Miller, 2003; Streichert et al., 2003; Capcarrere, 2004; Mange et al., 2000; Macias and Durbeck, 2002).

Natural organisms are often cited as examples of flexible, self-repairable systems and are often at the base of the works previously cited. Though this self-repair capacity seems to diminish with increasing complexity, examples of reconstruction are not rare in nature. Such capabilities rely on many different aspects, but (adaptive) growth, and especially re-growth, is certainly at the heart of these properties lacking so dearly in human artifacts. Growth is itself dependent on a fundamental structure of living being: cellularity. While this cellular aspect has been adopted in some of the previous studies, its necessity for self-repair has often been neglected. In earlier works (Righetti et al., 2003) however, we used only that cellular, decentralised aspect, without growth, as a means of getting fault-tolerant computation.

In this paper, we thus propose a simple cellular developmental system to gain robustness and self-repair capabilities. Rather than study growth in itself, as has been the case in most studies previously quoted, our aim here is to back the above-mentioned hypothesis about self-repair ability through experimental results. Nevertheless, we also demonstrate that by using a reasonably simple framework, it is possible to evolve practical solutions to two problems often confronted in works concerned with these issues. First we obtain stable growth. This may sound a very basic necessity, but is in fact extremely hard to obtain. Many of the previous works that studied growth highlighted that "cancer", uncontrolled growth is one of the main problems when developing these systems (Miller, 2003; Streichert et al., 2003; Capcarrere, 2004). In this paper, we show that by using a technical, practical, trick, it is possible to evolve perfectly "stable" growth, in the sense that perturbed system restabilises more often than not in their working state. Moreover we also evolve ideally decentralised computation, a task that has been shown to be non-obvious even for simple prob-

lems (Capcarrere, 2002).

In the first section we present the system framework and the evolutionary approach taken to obtain the results presented in the following section. These results, while not evolved specifically for self-repair, exhibit very good emergent robustness and self-repair capabilities. We then conclude on the many open paths for future research.

A Developmental Cellular System

While nature is the starting point for our inspiration, our purpose here is not bio-realism as is the case in (Streichert et al., 2003). The system developed for this work aims at extracting some quintessential abstract principles of the growth process while remaining computationally tractable. The eventual goal here is to gain fault-tolerance and self-repair capabilities in computing artifacts.

Our system can be divided into two main parts and we will describe these separately in the next two sections. The first part is what could be described as the *engine* or the physics of the system. It is directly inspired by the work of Miller (Miller, 2003) but is simplified. This is where the novelty of this specific system lies. The second part describes briefly the evolutionary algorithm used, and most importantly, the parameters used to allow for replication of the experiments.

The engine

The system developed and studied in this paper is based on the model of cellular automata. More exactly the cellular system lies on and develops along a two-dimensional discrete topology and each cell separetely executes the same program. According to its surrounding states and its own, the cell changes its state and decides whether or not to "replicate" in a free space of its neighbourhood.

Each cell lies at the vertex of a 2-dimensional lattice. It possesses a x-bit state that is readable *and modifiable* by any one of its 8 neighbouring cells, where x is dependent on the given problem. This neighbourhood matches Moore neighbourhood (North, N.-East, East, S.-East, South, S.-West, West, N.-West). Though not stored as a lookup table, the memoryless cell program will deterministically act according to the $9x$ bits input from all the neighbourhood cell states and its own. Like a classical CA, the result of this internal program execution will determine the next state of the cell. However, unlike the CA model, this cell may also alter the state of its 8 neighbouring cells. Hence this internal program could be described as a $9x$-bit input, $9x$-bit output function. It is important to note that while our model is close to the one developed previously by Miller, it does not include any chemicals nor any chemicals diffusion, and, is thus, in our opinion, simpler.

The grid used is non-toroidal. To the program, the "virtual" cells beyond the borders appear to have a 0^x state that cannot be altered. Experiments showed that this border is very useful to obtain a *stable* growth. Obviously it contains growth, but it also provides, unexpectedly, stability. The theoretical questions posed by the problem of "cancer" evoked earlier is not solved as such, but this border allows us to practically use the advantages of growth in terms of self-repair without its disadvantages.

The neighbour's state rewriting property obviously entails a series of questions on the order of update of the cells. It forbids a fully parallel update. A single model of asynchronous updating was studied. It is a fully sequential deterministic model where each cell is updated along a line sweep order (Giacobini et al., 2003). This is what was used in previous works (Miller, 2003). A more interesting model from a research viewpoint, but less practical from a hardware viewpoint, is a random asynchronous update mode. This is to be studied in future works. One time step of the simulation is the evaluation of n cells, where n is the number of cells in the environment.

The evolutionary framework

We used an adaptation of Cartesian GP (CGP). CGP was first designed to evolve digital circuits (Miller and Thompson, 2000) and is well suited for problems involving binary inputs/outputs. They present the great advantage of not suffering from the bloat problem and of keeping the representation compact. At the same time they allow for quick and efficient data manipulation and work with small populations. However, as a drawback, they do require a large number of generations, but one has to keep in mind that the number of sampled solutions is still extremely small compared to the size of the search space. We will not go into the details of the description of CGP here, but simply highlight the idiosyncrasies of the encoding, the selection and the genetic operator we used.

The encoding of our program is as described in (Miller, 2003). The inputs to the program, as highlighted in the above section, are the bits from

the states from all the cells in the neighbourhood. Each bit is considered individually as a possible input. We also use the same operations as were used in previous works by Miller. These consist of only four kinds of basic operations: if (input0) then output (input1) else output (input2); if ($\overline{input0}$) then output (input1) else output (input2); if (input0) then output ($\overline{input1}$) else output (input2); if (input0) then output (input1) else output ($\overline{input2}$). This allows for any boolean function of three variables. Each of these operators is to be called a node henceforth. The size of the evolved program is evaluated in terms of number of nodes.

The population size used in all experiments is only 5. Only the best individual is selected, and 4 new mutants are created from it to maintain the population size. Hence it is an extremely elitist algorithm. This is in great part imposed onto us by the very small population size. This was chosen as our experiments showed that reasonably bigger populations were not sensibly decreasing the necessary number of generations per run, thereby increasing greatly the time needed. It is highly probable that a significantly bigger population (5000 to 50000) would certainly greatly decrease the number of generations, but this has not been tested to date. Mutation is the only genetic operator used. It mutates an equal percentage of the following: Inputs to the CGP nodes, the function of the nodes, and the output table which is separate. The only thing worth mentionning here is that the mutator makes sure a cell gets its inputs from previous cells or system inputs, entailing a loop free, easily executable organism.

Emergent Robustness Through Growth

In this section, we present the results obtained. Very interestingly, we observe, as expected, that they exhibit *emergent* fault-tolerance through self-repair, thereby entailing lasting safe behaviour. Two experiments are described here: – first, an experiment that displays visually this emergent self-repair property; – second, a functional experiment, surely more promising in terms of potential future research and applications.

A "visual" example

The first set of experiments aims to demonstrate the qualities of robustness and self-repair obtainable in growing systems. To do so "dramatically", we chose a morphogenetic example where the reconstruction process is visually interesting. The aim of the task is to get a system that grows *and stabilises* in the shape of a French flag, i.e., a rectangular shape, made up of three different kinds of cell, red, blue and white, themselves arranged in rectangles. As the growth process is not our main focus for this research, we chose to use a shape that has been shown to be evolvable in the past (Miller, 2003).

We use an 8x8 non-toroidal environment in which we aim at evolving systems that develop into a 6x3 horizontal shape, made of 3 blocks of 3x2 cells each. Each cell has got a 5-bit state, the first two bits encoding the colour displayed. The aim is to get a stable configuration in which the first two bits of the cells' state are: 00 (encoding black) outside the rectangular shape, 01 (blue) for the cells in the first block, 10 (white) in the second block and 11 (red) in the third block (see fig 1). The workings of the system are no different than the model described in the above section but there is a specific constraint on the internal workings of the program. If the first two bits of the state of the cell are 00 then that cell's program does nothing. The interpretation of that constraint could be that a cell with a 00-starting state is dead, or environmental, and the other cells are alive. Each cell being able to change its own and its surrounding states, it can thus "decide" to die, reproduce or kill a neighbouring cell.

The CGP evolutionary algorithm used was as described earlier with a population of 5 individuals. The starting configuration is an empty grid, all cells with 00000 state, except for the central cell's state set to 10000. Fitness at time step t, f_t, is defined as the number of cells whose first two bits are in the expected final state. The other three bits which may be, and are, used for the computation are not evaluated directly in the fitness. The fitness used for the evolutionary purpose, F, is defined as the simple aggregate of the fitnesses after 7 and 15 time steps, $F = \frac{f_7+f_{15}}{2}$. The two fitnesses are employed to discard any unstable growth strategy. The hurdle of unstable and unlimited growth highlighted earlier has been solved here using *both* this fitness constraint and the non-toroidal environment described earlier.

Successful evolution of "organisms" growing and stabilising in the desired shape took on average 300000 generations. Perfect (64/64 cells) or excellent individuals (63/64 cells) in terms of fitness were obtained on 54 out of 100 runs. Though growth is not the focus of this paper, it may be noted that these results compare favourably with past studies,

both in terms of success rate and on the stability of the growth of the evolved organism. The size of the successful organisms' program uses less than 100 nodes, which is reasonably small. Further investigations are necessary however to determine exactly how many of these were necessary for the task, as "junk DNA" constitutes, a priori, a non negligible part.

Emergent robustness: The purpose of this paper is to back the hypothesis that cellularity and growth are sufficient conditions for robustness and self-repair. Therefore we tested out the solutions evolved in a *safe* environment. It is important to note that the self-repair capability is thus *emergent* and as such can be deemed to be a consequence of the developmental and cellular nature of the system. The error model used here is a one-time strike model. From the seed configuration, the organism is run for 20 steps. After 20 time steps, an error occurs, once, and the system is run again for 100 more time steps. The experiments described below are on one specific individual but our tests, still preliminary, seem to show that these properties can be roughly found in the other organisms successfully evolved.

Experiment 1: After 20 steps, one of the cells of the grid is reset to 00000, i.e., both the visible and invisible bits are reset. One should note also that this "kills" the cell subject to the error and thus it cannot repair itself. In 95% (61/64) of all the possible cases, the organism perfectly recovered, reconstituting to its perfect working state.

Experiment 2: In this experiment, 10 cells at random are reset in one go. Obviously it is impossible to test the behaviour exhaustively on all C_{64}^{10} possible cases, but the experiment was repeated randomly 1000 times, giving a good overview of the behaviour of the system in the case of dense, but non block, errors. The system resisted rather well and reconstituted itself perfectly in 65.4% of the tests. An example of such a perfect recovery may be seen in figure 1. The system reconstituted itself but for one cell in a further 18.8% of the case.

Experiments 3,4,5: In these experiments, we test block rather than sparse errors. We reset, respectively, all the cells in the white block, red block and blue block. In each of these experiments the system recovers completely in less than 10 time steps. In a *sixth experiment*, we removed both the red and white blocks at the same time, and the system still recovered completely.

However, the system is not indestructible and

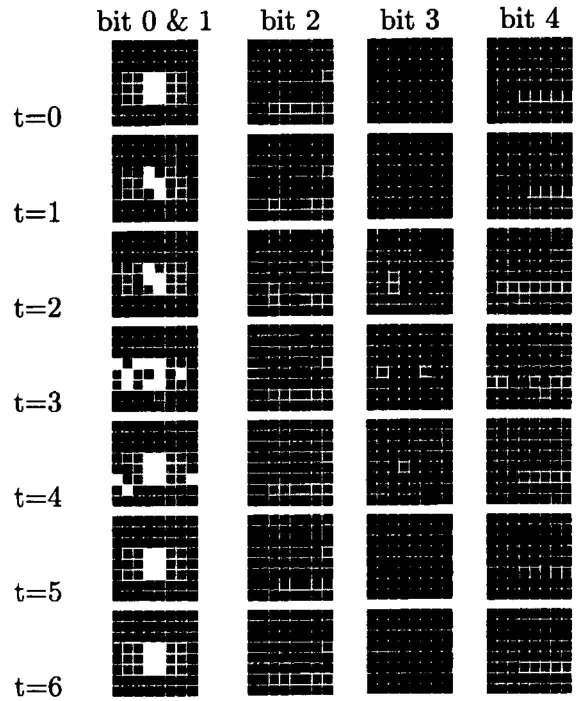

Figure 1: The state of the system for 7 time steps. The growth phase is not shown here, and at time 0 we see the system in its mature, stable state. At time 1 the errors occur. Time 2 to 6 shows the recovery. Time flows downward. Horizontally one can see the state of the two first bits of each cell (the "visible" state), and then of each of the following bits of its state.

when the error block is too large or of horizontal shape the recovery is mitigated or even impossible.

In *experiments 7 and 8*, respectively, the red and white block, and the blue and white block is removed. The system does reach a stable state and a complete French flag is reconstituted, but spurious visible states remain and the original state is never recovered. Hence the self-repair process cannot be judged satisfactory.

Finally, in *experiments 9,10,11*, respectively, the horizontal top, middle and bottom line is removed. The middle line removal causes no trouble, and the system fully recovers. The top and bottom line, on the other hand, creates total chaos, and the system only recovers pieces of the French flag shape, and its stable state includes lots of spurious cells. The recovery in these cases is beyond its ability.

As can be seen, while not perfect, the emergent self-repair capabilities in this visual example are striking. This should be nuanced by the fact that our error model, though harsh in its extent in space, is rather kind time-wise. This aspect will be investi-

gated in future studies. Nevertheless the self-repair process is very efficient and takes usually less than 10 time steps. Hence, the assumptions behind a strike-once model are not unreasonable.

If self-repair capabilities are to be of any use, it should be applied to a functional rather than a visual example. This is investigated in the next section.

A "functional" example

As the French-flag example illustrated, systems based on cellularity and development may display interesting self-repair capabilities. To go beyond this "dramatic" example, we explore the possibility of making a digital circuit on a cellular structure configured through a developmental process and then test its self-repair capabilities, if any. The idea to have a cellular, developmental digital circuit to increase robustness is not new in itself (Mange et al., 2000). However, our system is simpler in its deployment as it can be evolved. More importantly, it allows at the same time for "automatic" error detection (no special signal is needed), and recovery from more bulky errors than the embryonics system, developed by Mange *et al.*. Of course, there is a price to pay, which is the lack of guarantee, before testing, on the exact recovery abilities. Another interesting approach is the cell matrix approach (Macias and Durbeck, 2002) which is rather close to our approach of self-assembling and reconfiguration. Besides, their real hardware approach is certainly interesting for future applications. However, unlike our system, it requires active detection of errors and must be hand designed.

We adopted a very abstract approach. Our circuit is made-up of a grid of $n \times n$ multiplexers, topologically identical to our $n \times n$ cell grids. Each cell's state encodes which type of multiplexers is to be at this position and and what are its connections. The states of all the cells could be interpreted as the configuration string of a basic homogeneous FPGA. All the external inputs of the system are connected by default to false, except for two (or more) of them which are connected to the inputs of the problem. Identically, one output (or more) is designated as the output of the system. The resulting circuit can then be evaluated to see if on given inputs it gives the expected output(s).

The first two bits of the state of the cell determine which of the four types of multiplexer is used. All the multiplexers have three inputs and one output. The first input determines which of the second or

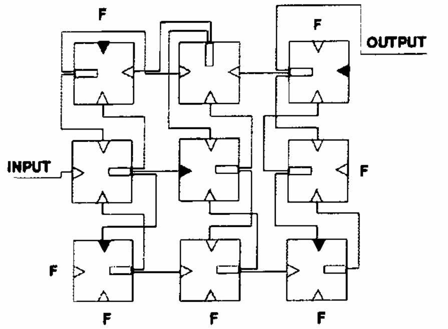

Figure 2: Example of a 3x3, one-input, one-output circuit with its connections. The triangles on the multiplexers are its three inputs, and the bar is its output. Black inputs are inverted. The evalutation of the circuit is done right to left, top to bottom 4 times before the global output of the circuit is read.

third input is connected to the output. In the first type no input is inverted, in the second type input 1 is inverted, in the third input 2 is inverted and finally in the fourth input 3 is inverted. The multiplexer output is always potentially connected the 4 adjacent multiplexers, however it can have four directions, each one being the 90 degrees rotation of the previous, which will determine which multiplexer is connected to which input (see Figure). The next two bits of the cell's state encode this. One may note that this may create meaningless circuits. Being in an abstract model, this is not a problem, and the circuits not working are naturally eliminated from the population evolved. There are a further three bits available in the cell's state that are not used to directly code the multiplexers configuration. The evolutionary process used is exactly as in the previous experiments. The fitness here is quite simply how many inputs to the circuit give a correct output. Thus, unlike in the previous experiment, there is no direct constraint on the cells' state from the fitness. This is a rather crude fitness, but that proved sufficient to evolve successful system. It is important to note that unlike the last experiment in this set of experiments all cells are "alive", i.e., the internal program is executed in every cell, whatever its state.

Results: Using this scheme, it was possible using 5×5 grids to evolve AND and XOR gates. In both cases the inputs were connected to cells (0,1) and (0,3), and output to cell (4,1)[1]. Obviously this

[1] Each multiplexer in the $n \times n$ grid is numbered by a pair (i, j), where i is the column number and j, the row number

scheme uses a high redundancy of resources. However it should be noted that this increase in the size of the search space does not increase the complexity of the evolution. Actually, basing ourselves on early experiments, we can even say that it simplifies the evolutionary process greatly. We obtained perfect evolution in 100% of the runs. Here the 100 node CGP cell's program was evolved in less than 5000 generations on average, two orders of magnitude less than for the French flag experiment. Though our work is still in an early stage, we were able to evolve also more complex systems, such as a full binary adder using a 9×9 grid.

Emergent robustness: The XOR gate was tested for robustness using the same error model as for the French flag example. While it is obvious that is extremely redundant to use a 5×5 grid to make an XOR circuit, it is important to note that this is *not* what makes the circuit fault-tolerant. As for the French flag, after an error occurred, the circuit reconstitutes itself, i.e., reconfigure itself to exactly the original configuration. Hence all of the 25 multiplexers are of the "correct" type in the "correct" position, even though some of them are not necessary to the workings of the circuit. In a first experiment, a unique cell is reset after 20 time steps. The circuit recovered fully and completely in all the 25 cases possible. The self-repair ability seeming good, the circuit was further tested with a 5 random cell error. It recovered fully in 70.8% of 1000 experiments. The error was only on one bit of only one cell's state in a further 23.2% of the experiments. And the biggest error was no more than 7 bits out of the 175 (25×7) bits of all the states of the cell. This very good recovery property of this circuit is further increased if one takes only into account the functionality of the circuit, and thereby uses its inherent redundancy. Then the XOR circuit gives the correct output in 100.0% of 1000 5-cell error and 1000 10-cell error experiments.

Concluding Remarks

In this paper we have shown that it was possible to evolve systems exhibiting self-repair capabilities thanks only to their workings. The only direct constraints on the system are that it should have a cellular structure and should develop into its desired final state. The "error-detection" and self-repair properties are the emergent consequence of these constraints. These encouraging results, most notably the fact that perfect repair occurs in a vast majority of cases, call for more research in these directions.

Obviously the functionalities explored were still basic, and the error model used, while not unrealistic, is rather kind. Future works should explore further error models and aims at more complex tasks. Attention should also be devoted to understanding the workings of the developmental process so as to establish clearly the limits of the repair abilities.

References

[1] Capcarrere, M. S. (2002). *Cellular Automata and Other Cellular Systems: Design & Evolution.* Phd Thesis No 2541, Swiss Federal Institute of Technology, Lausanne (EPFL).

[2] Capcarrere, M. S. (2004). An evolving ontogenetic cellular system for better adaptiveness. *BioSystems*, (To Appear).

[3] De Garis, H. (1999). Artificial embryology and cellular differentiation. In Bentley, P., ed., *Evolutionary Design by Computers*, pages 281–295, Morgan Kaufmann.

[4] Giacobini, M., Alba, E., and Tomassini, M.. (2003). Selection intensity in asynchronous cellular evolutionary algorithms. In et al., E. C.-P., ed., *GECCO 2003 proceedings*, vol. 2723 of *LNCS*, pages 955–966. Springer-Verlag.

[5] Macias, N. J. and Durbeck, L. K. (2002). Self-assembling circuits with autonomous fault handling. In Stoica, A., Lohn, J., Katz, R., Keymeulen, D., and Salem Zebulum, R., editors, *Proceedings of the 2002 NASA/DoD Conference on Evolvable Hardware*, pages 46–55.

[6] Mange, D., Sipper, M., Stauffer, A., and Tempesti, G. (2000). Towards robust integrated circuits: The embryonics approach. *Proceedings of the IEEE*, 88(4):516–541.

[7] Miller, J. F. (2003). Evolving developmental programs for adaptation, morphogenesis, and self-repair. In W. Banzhaf et al, ed, *ECAL'03 proceedings*, vol. 2801 of *LNAI*, pages 256–265, Springer-Verlag.

[8] Miller, J. F. and Thompson, P. (2000). Cartesian genetic programming. In *EuroGP'00 proceedings*, vol. 1802 of *LNCS*, pages 121–132, Springer-Verlag.

[9] Righetti, L., Shokur, S., and Capcarrere, M. S. (2003). Evolution of fault-tolerant self-replicating structures. In W. Banzhaf et al,, ed., *ECAL'03 proceedings*, vol. 2801 of *LNAI* , pages 278–288, Springer-Verlag.

[10] Streichert, F., Spieth, C., Ulmer, H., and Zell, A. (2003). Evolving the ability of limited growth and self-repair for artificial embryos. In W. Banzhaf et al, ed., *ECAL'03 proceedings*, vol. 2801 of *LNAI*, pages 289–298, Springer-Verlag.