

# Evolving Genetic Regulatory Networks for Hardware Fault Tolerance

Arne Koopman<sup>1</sup> and Daniel Roggen<sup>2</sup>

<sup>1</sup> Adaptive Intelligence Laboratory, Intelligent Systems Group,  
Institute for Information and Computing Sciences, Utrecht University, The Netherlands

<sup>2</sup> Autonomous Systems Laboratory  
Institute of Systems Engineering, EPFL, Lausanne, Switzerland  
acmkoopm@cs.uu.nl

**Abstract.** We present a new approach that is able to produce an increased fault tolerance in bio-inspired electronic circuits. To this end, we designed hardware-friendly genetic regulatory networks based on a bio-inspired hardware architecture called POEtic tissue. To assess its preliminary functionality, the parameters of the genetic regulatory networks were evolved using genetic algorithms to achieve elementary behaviours, including patch growth and oscillations at various frequencies. The tolerance to faults was explored by inflicting several types of damage, and results show that the system exhibits capabilities to recover from them.

## 1 Introduction

Nothing lives forever. However, when we look at nature we see that when parts of an organism fail to operate, other parts may adapt to recover its functionality. This is not at all the case for electronics, as failing sub-circuits usually lead to overall disrupted operation. This paper discusses a new approach based on a bio-inspired technique to gain fault tolerance in electronic circuits. More specifically, it presents a hardware-friendly genetic regulatory network (GRN), able to grow structures based on artificial proteins and genomes.

The system is targeted for a new bio-inspired hardware chip called POEtic tissue, able to implement the three distinguished ways of biological adaptation: evolution, growth and learning [6]. Each chip can be used to implement a particular combination of these adaptation methods. In the work presented here, multi-cellular tissues are created from a collection of identical cells, which differentiate to attain specific functions based upon genetic information and inter-cellular signalling mediated by artificial proteins. The artificial proteins determine which genes are expressed, and in this fashion, the static genetic information is able to generate dynamic patterns, as proteins continuously arise, diffuse, and decay. These dynamic properties may improve fault tolerance of electronic based circuitry, as damage may be counteracted by protein production. We induce faults during the evolution of the genetic regulatory networks, to see how resilient this method is to damage.

## 2 Genetic Regulatory Network Cells

The multi-cellular system consists of totipotent cells, each of them having the capability to replace another due to an identical 'blue print'. Each cell has an identical ge-

nome combined with a unique protein mixture, the proteome, which determines its cell type. Implementation is done by means of POEtic "molecules": small FPGA hardware elements that can be configured to perform elementary arithmetic operations. The resulting tissue aims at creating bio-inspired evolvable hardware, which is not restricted to amplifiers or similar typical electronic blocks, but extends to higher level applications like neurons or oscillators. As a whole, it can be considered as a PO model, a combination of evolution (phylogeny) with growth (ontogeny). On top of this an epigeny layer can be added, for example in the form of neurons, although this is not addressed in this paper [5].

Each cell contains a genetic regulatory network (GRN). In a GRN, artificial genomes are divided into genes that can be unlocked when cell environmental requirements are met. This locking and unlocking of genes is inspired by the transcription of biological genetic material, and is initiated by the binding of proteins onto specific gene related regions, called *cis* regions. Connected proteins can either initiate or cease protein production, called expression or inhibition respectively. This relation is encoded in the gene: it encodes which protein type can bind and which protein type is produced. The complete system is situated in a simulated environment in which proteins can exist in various concentrations within the cell. These protein concentrations have to be beyond a genetically encoded threshold for the gene to be 'unlocked'. Since the proteins in the system act upon the genome, the genome is paired with the proteins within the cell. Multicellular GRNs contain multiple copies of identical genomes and allow proteins to be diffused between neighbouring cells (4 immediate neighbours). Different diffusion rates of the related proteins that flow from cell to cell give rise to variations in viscosity. Moreover, protein types in a system may vary in their half-life (decay rate), which limits the temporal effect of produced proteins. In this sense the system may exhibit temporal behaviour due to production and decay of proteins, while diffusion may be used to generate spatial patterns.

### 3 Implementation

Most simulations utilising GRNs have almost no restrictions laid upon them, in contrast to designing hardware friendly versions [2]. Minimal size constraints and computational tractability are key-points of the minimalist GRN system presented here [3]. All parameters have been encoded in 7 bit values and are manipulated by fixed point arithmetic. The GRN mechanism is divided into three separate pathways: protein decay, diffusion and production.

Digital hardware systems don't allow real valued chemical diffusion; therefore proteins diffuse according to integer protein gradients among neighbouring cells. Diffused proteins to neighbouring cells are added to their proteomes. Cells located at the borders are connected to the other side of the tissue, making it boundless for the diffusion mechanism. Produced proteins arise from the regulatory pathway which uses a specific genome encoding. In this work we will discuss two hardware friendly encodings from which one was selected for implementation in hardware.

#### 3.1 Minimal Level Encoding

Four main regions make up the gene. Genes are activated when a particular protein, encoded in the *cis* region, is available in the proteome of the cell. Upon activation the

gene produces a protein type encoded in the second field,  $prod$  (see fig. 1b). Not only the mere absence or presence of the proteins is of importance: another field includes the minimal protein concentration,  $min$ , for the gene to become active. This relation can be flipped by a bit,  $i$ , which determines whether the gene is an excitatory or inhibitory gene. The former starts producing proteins ( $P_{prod}$ ), dependant on how far the level is exceeded, the later slowly diminishes production as the protein concentration of type  $cis$  ( $P_{cis}$ ) reaches the encoded level (Eq. 1). Note that due to hardware friendliness, this relation is linear. In total each gene of this encoding consists of 12 bits, 2 for the  $cis$  and  $prod$  field each, 1 bit indicating its behaviour and 7 remaining bits encoding the minimal level.

$$P_{prod} = \begin{cases} \max(min - P_{cis}, 0) & i = 0 \\ \max(P_{cis} - min, 0) & i = 1 \end{cases} \quad (1)$$

### 3.2 MinMax Level Encoding

While the former encoding is still quite biological plausible, this encoding adds another level which is unseen in nature. This extra level,  $max$ , encodes the concentration at which the behaviour of the gene switches back (see fig. 2c bottom). In the excitatory case, the exceeding of this maximum level leads to no proteins generation. In the inhibitory case, proteins gradually arise when biased to this maximum level (Eq. 2). The addition of this extra 7 bits level makes it a 19 bits gene.

$$P_{prod} = \begin{cases} \max(min - P_{cis}, 0) & i = 0 \ \& \ P_{cis} < min \\ \max(P_{cis} - max, 0) & i = 0 \ \& \ P_{cis} > max \\ \max(P_{cis} - min, 0) & i = 1 \ \& \ P_{cis} < max \end{cases} \quad (2)$$

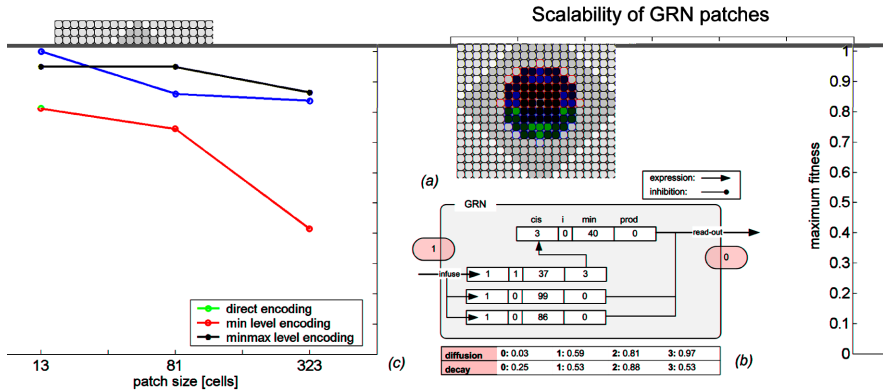
### 3.3 Hardware GRN Cells

The simulated GRN model is used to derive experimental results and is translated in POEtic hardware molecules, for use in future intrinsic EHW experiments. The current serial implementation of the min level GRN cell is about 200 molecules (elementary functional units). Estimates of the final POEtic chip also lead to approximately 200 molecules, meaning that one cell could be implemented in one chip.

Larger cell grids can be created by combining several chips together, an inherent property of POEtic tissue. The exact size depends on the amount of genes in the cell, as they are included in a separate memory module of the system. The hardware implementation operates on one gene at a time. As genes reside in memory cells, memory extensions easily lead to additional genes. The current design is targeted for 4 proteins, extending this requires a redesign of the implementation. Given the fact that our minimal level encoding seemed to perform quite well in simulation, its low space requirement made it a logical candidate for hardware implementation.

Inside the cell a table contains the protein decay rates for all available types. In the decay pathway, each entry in the protein table is cycled through and scaled by this value. In the regulation pathway, the  $cis$ -region produces an address for a selectable memory cell; this outputs the current concentration, which is compared by a switchable comparator that receives the min level and an inhibition bit as an input. Its output, the biased value, is added to the temporary protein table, which points to the

entry directed by the production region in the gene. Since this pathway expresses only one gene at a time, all genes are cycled through before proceeding to the next pathway. The diffusion pathway involves the communication between neighbouring cells forms the most intricate and space consuming pathway. Each current protein concentration has to be scaled by diffusion rates stored within a table and transmitted to neighbouring cells. At the same time it has to process received concentrations to the current concentrations.



**Fig. 1.** (a) A grown patch on simulated tissue. (b) The evolved GRN that generates the patch. (c) Maximum obtained fitness for different encodings: direct, min- and minmax level GRN, after 200 generations and averaged over 5 runs [3].

## 4 Applications

Genetic regulatory networks have been mostly used for creating systems that can grow and develop structures in space [1]. Since the proteome in a single cell behaves dynamically due to infused proteins from neighbouring cells and external injections, it is also possible to generate spatial-temporal patterns [4]. Furthermore, this dynamic nature could also make the tissue somehow fault tolerant. As functionality is not directly encoded in the genome, but comes from the interaction of produced proteins, complete structures and functionality could be more stable when parts of the tissue fail to operate properly as they may be compensated by internal dynamics.

Before looking at fault tolerance, we address the basic capabilities of the GRN tissue. First of all, its ability to grow basic structures in fault-free conditions is compared to a direct encoding. As fault tolerance in our case is related to the ability to utilise the dynamic behaviour, the next series of experiments focuses on oscillation capabilities. Successfully evolved behaviour suggests that the system can produce dynamics to compensate for induced damage.

In the experiments, specific proteins are infused in the GRN tissue in the first clock cycle after which the tissue operates without external influences. The growth period is defined as the number of cycles the GRN system is run. During the growth period, normally 40 cycles, the tissue is evaluated by the fitness function, and results in an averaged fitness value at the end of the growth period. This value is used by a genetic algorithm operating with truncation selection (25%), 1 point-crossover (50%) and a

uniform mutation rate of (1%). Genome lengths vary upon the task and are measured in amount of genes, each either 12 or 19 bits long, dependent on the used encoding scheme. Protein specific parameters are also encoded within the genome. For each four protein types, two specific parameters defining protein diffusion and decay are encoded each on 7 bits.

#### 4.1 Patch Growth

The objective of this task is to grow patches of specific target sizes (ranging from 13 to 323 cells), which is initiated by a single protein infusion in the centre of the tissue (fig. 1a). Next to the GRN encodings a direct encoding was used to compare the scalability of the encodings, in which case each bit directly encodes the state of the cell. Maximum fitness is obtained when exclusively the cells within the target area contain the required target protein type 0 (Eq. 3&4). Figure 1 shows that direct encodings compete quite fairly to the developmental encodings on a 13 cell patch (radius is quarter of tissue size), but loses some performance when this patch is scaled relatively to 81 cells with a larger tissue size of 20 by 20 cells. A patch of 323 cells (radius of half the tissue size), shows that the minimal level encoding starts to degrade, this in contrast to the minmax level encoding (see fig. 1 c).

$$f_t = \frac{COUNT_{IN}}{patch\ size} \cdot \frac{COUNT_{OUT}}{tissue\ size - patch\ size} \quad (3)$$

$$fitness = \frac{1}{growth\ period} \sum_{t=0}^{growth\ period} f_t \quad (4)$$

When we look at the evolved GRNs, we see that the stable equilibrium state is obtained due to the decay and diffusion parameters; proteins are precisely triggered to vanish due to decay at the edge of the patch (see fig. 1 a). Differently sized patches can be the result of different decay and diffusion rates. The better performance of the minmax level encoding in the relative larger patch may be explained by the fact that this encoding can trigger also on a maximal level, so the 'edge' of the regulating protein does not have to be as sharp as with the minimal encoding. At the same time, this seems to force the min level encoding to grow at a slower pace.

#### 4.2 Oscillation

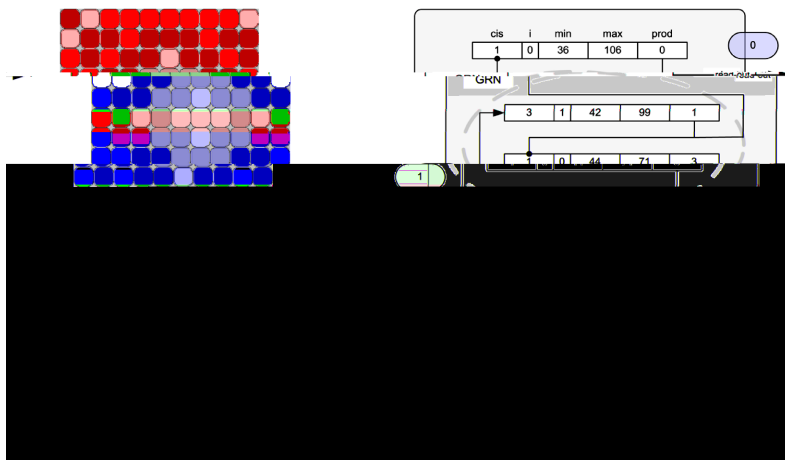
Initial experiments showed that the genetic regulatory system automatically produces temporal behaviours (which resulted in spatial checkerboard patterns as well as oscillating cells). While these auto-oscillations contain high frequencies, it is interesting to see whether evolution can fine-tune oscillation to a specified oscillation frequency. Fitness is obtained for minimising the difference, *diff*, between a protein concentration,  $P_o$ , and a target sine of a given period (Eq. 5&6).

First we infuse a maximum quantity, *max*, of proteins at time step 0 in the centre cell; we look solely at one cell to determine its fitness. Although in practice it is shown that neighbouring cells mimic the oscillating behaviour of the centre cell, leading to growing circles that emerge from the injection spot (see fig. 2a).

$$diff = \left| \frac{max}{2} + \frac{max}{2} * \sin\left(\frac{t}{period}\right) - P'_o \right| \quad (5)$$

$$\text{fitness} = 1 - \frac{1}{\text{growthperiod}} \sum_{t=0}^{\text{growthperiod}} \frac{\text{diff}}{\text{MaxConcentration}} \quad (6)$$

We want to assess the ability of the tissue to produce oscillatory behaviour at different frequencies. To this end we swept the target oscillation period between 4 and 20 cycles (see fig. 2b). Environmental conditions were slightly altered by adding one gene in the genome, leading to 5 genes. Both encodings successfully evolved the desired oscillation frequency when we look at the maximum fitness of 5 equal runs (see fig. 2b). This freedom to oscillate at different frequencies can be attributed to the evolvable decay rates, which can fine-tune the decay for it to reach a critical level near the beginning of the oscillation period. In the evolved individuals 3 out of the 4 proteins have been exploited by evolution



**Fig. 2.** (a) Growing rings emerge from an oscillating cell. (b) Various target frequencies are equally evolvable. The shown results are maximum obtained fitness values derived from the minmax encoding after 5 averaged runs of 200 generations. (c) 2 genes in the GRN interact with each to produce oscillations [3].

The evolved GRN has 2 main operating genes that produce the oscillatory behaviour (see fig. 2c). The first gene, *gene 0*, (seen bottom in fig. 2c) produces protein 3 in the presence of protein 1. The lack of protein 3, due to decay, activates the second gene, *gene 1*, (seen middle in fig. 2c), and starts to produce protein 1. This in effect leads to intertwined genes, in which each stimulates the other. The remaining genes in the network are evolved as read-out genes, and map protein type 1 to the target type 0.

## 5 Fault Tolerance

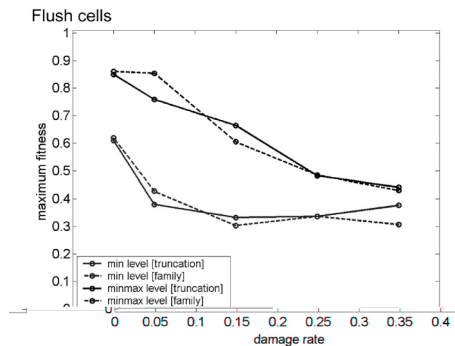
We can inspect the fault tolerance of a system in various ways. To our knowledge GRNs have never been used in fault tolerance tasks. We chose to damage aspects directly related to the gene regulation process. First of all, we flushed the protein concentration, similar to a memory, of random cells and at random time steps during the growth period. A second type of failure was induced by stopping the gene regula-

tion process of random cells, but allowing proteins to be infused into the cell and thus indirectly determine the proteome of the cell. Finally, we looked at how the system performs when cells are killed; no input and output of any kind was allowed, and no regulation occurs in the cell, making it a dead cell in the tissue.

We applied all of these types of fault injuries to both developmental encodings and rerun the experiments 5 times, each lasting 200 generations. Environmental conditions are similar to the previous task: a growth period of 40 cycles, 5 genes in each genome, 4 protein types available in the system and the tissue consists of 10 by 10 cells.

## 5.1 Flush Cells

During the whole growth period of the individual, each cell has a certain probability to have its proteome flushed. Such error is random in both time and space, and can be conceived as a memory reliability error. Although both encodings seem to handle minor injuries, we see that the performance of the min level encoding drops sharply when the flush damage rate is increased (see fig. 3). The performance stabilises above a 10 percent damage rate, but it has to be noted that the corresponding behaviour is quite erratic. Performance degrades more gradually for the minmax level encoding, which still performs reasonably well at the 20 percent damage rate. In contrast to the fault-free evolution, we observe that proteins are evolved to be more dynamic (higher diffusivity and decay). More dynamic proteins in combination with a production interaction can counteract the loss of proteins more easily.



**Fig. 3.** Maximum fitness values averaged over 5 runs, when flushing the proteins during the growth period at various damage rates.

## 5.2 Freeze GRN Regulation

At the first time step of the growth period, random cells have a probability to lose their ability to generate proteins in their subsequent life time. Note that these damage probabilities are incomparable to the former ones. Damage is only induced during the first cycle and kept spatially constant, which may be easier for the GRN to handle.

When applied to both encodings we see that the min level encoding performs worse with low damage rate (see fig. 5 left). A possible explanation is that min level

encoding only has one boundary region, and thus protein level have to be sharp decaying at the edge of the patch. This suggests that for low damage rate the production of proteins has to be “tamed”, to limit the growing patch. As fitness is measured during the whole growth period, this slower growing patch leads to less fitness. This is supported by the evolved diffusion and decay constants; which are less dynamic in fault free conditions than in the fault induced case (see fig. 1b & 4). Resulting, we see that the minmax level encoding performs slightly better than the min level encoding. Moreover we also performed the identical injuries on patches grown by a GRN evolved in fault free conditions (dashed lines are single runs). We see that performance decreases considerably, suggesting that the fine-tuning of protein properties trough evolution is beneficial. We also verified the same behaviour on a patch 6 times bigger, in which a similar response is shown, but starts to degrade at 10 percent lower damage rate.

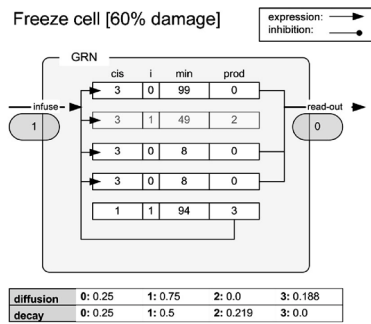


Fig. 4. An evolved GRN that grows a patch in freeze regulation conditions at a 60 damage rate.

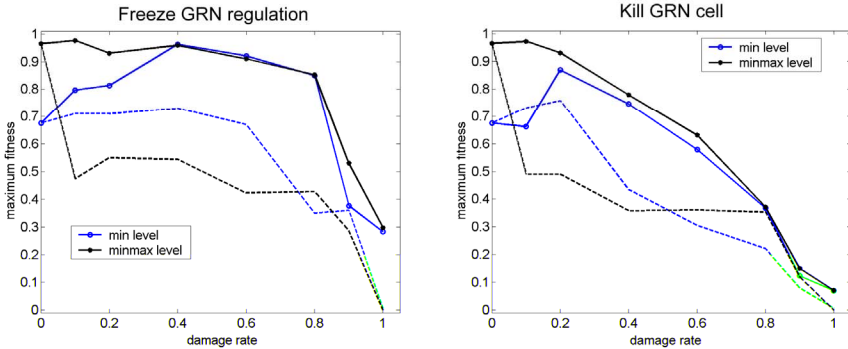
### 5.3 Kill Cells

While the former experiments only disabled the decoding mechanism at the beginning of the growth period, we now disabled random cells completely. We see a linear decrease of performance when we increase the damage rate (see fig. 5 right). This is comparable with a non adaptive technique in which damage directly causes cell malfunction. Also in a system using a direct genetic encoding, performance would degrade also linearly. As in the freeze regulation we induced the same type of damage on a GRN evolved in fault free conditions. Like previously, we see that evolutionary pressure on the GRN and its diffusion and decay constants is beneficial for system performance, which is indicated in the right side of Figure 5 (dashed lines are GRN evolved in fault free conditions).

## 6 Discussion

In this paper we described results derived from a hardware-friendly multi-cellular genetic regulatory network which has been implemented on a new bio-inspired electronic circuit called POETic tissue. Simulations were conducted to derive experimental results before committing to the hardware implementation. As fault tolerance is one of

the main topics in current bio-inspired electronic architectures, we explored the capacity of the GRN to withstand faults and recover from them. Thanks to gene reuse, GRN may scale better than direct encodings when the phenotype size increases. We compared a direct encoding to the two developmental encodings: min- and minmax level encoding. Minmax level encoding performs better compared to the direct encoding. The comparatively slower growth of the patch when using the min level encoding, leads to a lower fitness. If damage has to be counteracted by internal dynamics, evolution has to be able to tame the dynamics of the GRN. Oscillation is an obvious task to measure temporal capabilities of a GRN. Both GRN mechanisms handle various target oscillation periods well, thanks to the evolvable decay and production constants.



**Fig. 5.** Maximum fitness (5 runs, 200 generations) when freezing the regulation (left) or killing the cells (right). The evolved GRN handles quite some damage during regulation freezing. Damaging a normal fault free evolved patch leads to a drop in performance (dashed).

The two GRN mechanisms behave differently to faults. Both can sustain slight damage when faults consist of flushing the proteome. However, the minmax level outperforms the min level encoding and obtains a high fitness even at a considerable damage rate. The difference between both encodings also becomes apparent when freezing the regulation process. The min level encoding needs a limited growth speed in order to grow the specified patch. An increased damage rate allows a quicker growth and thus leads to a higher average fitness. Patches stay quite stable until damage reaches 80 percent and stability of the phenotype collapses. However, the GRN systems are more tolerant to faults than direct genetic encodings and show a less than linear decrease of the fitness with an increase in damage rates. When killing cells, the performance degrades in a linear fashion. This is normal as the tissue is considered irreversibly damaged. However if evolution occurs with faults, it manages to adapt to such condition to minimize the performance degradation, in comparison to evolution without faults. Although these results seem promising, they come at a high cost. The current hardware implementation uses about 200 molecules, which corresponds to a full POetic chip. Developing a more compact implementation is the first step to create more realistic experiments. The ease at which these GRNs could be evolved may indicate a fruitful application in future work on more complex tasks like robotic control in extreme conditions.

## References

1. Eggenberger, P. & Dravid, R.: An evolutionary approach to pattern formation mechanisms on Lepidopteran wings. In the Proceedings of CEC1999, pp. 470-473 (1999).
2. Gordon, T.: Exploring models of development for evolutionary circuit design. In CEC2003, the Congress on Evolutionary Computation, pp. 2050-2057 (2003).
3. Koopman, A.C.M.: Hardware-Friendly Genetic Regulatory Networks in POEtic tissue. M.Sc. thesis. Institute for Information and Computing Sciences, Utrecht University (2004).
4. Quick, T. et al.: Evolving embodied genetic regulatory network-driven control systems. In Proceedings of the Seventh European Conference on Artificial Life, ECAL'03 (2003).
5. Roggen, D., Floreano, D. & Mattiussi, C.: A Morphogenetic Evolutionary System: Phylogeny of the POEtic Circuit. In Proceedings of the Fifth International Conference ICES 2003, Evolvable Systems: From biology to Hardware. 153-164 (2003).
6. Tyrell, A.M., Sanchez, E. Floreano, D. Tempesti, G. Mange, D. Moreno, J-M. Rosenberg, J. & Villa, A.E.P.: POEtic Tissue: An integrated architecture for bio-inspired hardware. In Evolvable Systems: From biology to Hardware. Proceedings of ICES 2003, pp. 129-140, Springer-Verlag (2003).