Obtaining System Robustness by Mimicking Natural Mechanisms

Song Zhan, Julian F Miller, and Andy M Tyrrell

***Abstract*—Real working agents normally operate in dynamic changing environments. These changes could either affect the efficiency of the agents' performance or even damage the functionality of the agent. Robustness is the key requirement to solve this problem. Inspired by nature, this paper demonstrates two mechanisms that contribute to individual's robustness in changing environments: evolution and degeneracy. Through evolution in damaging environment, evolved agents have to cope with changes in the environment and acquire robustness. Through degeneracy, individuals can maintain their fitness even when some damaged parts are involved in system function.**

I. Introduction

ROBUSTNESS describes a system's ability to maintain functions with perturbations [6] and is a desirable characteristic for any system working in changing or damaging environment. Many mechanisms have been investigated to assist the achievement of robustness such as modularity, degeneracy, redundancy and distributed robustness [3]. These mechanisms are present in nature. In this paper, we focus on two of these aspects: evolution and degeneracy. Our approach is based on a biologically inspired self-organization system for the construction of electronics circuits that show characteristics of robustness. Section 2 introduces the model. Section 3 describes the techniques devised for achieving robustness. Section 4 describes the experiments. Section 5 presents the results and the analysis. Section 6 concludes the paper.

II. The Model

The experiments shown in this paper are based on a biology-inspired model of a developmental gene regulation network that has been described in more detail in [7, 8] .

A. Genetic Representation

DNA consists of genes and these genes have regulatory regions and product regions. Regulatory regions are used for self-regulation and feedback and also could be coupled with environmental signals. We have distinguished two types of regulations in our model: one is for enhancing gene action and the other is for inhibiting gene action. Figure 1 is the example of one gene where 0 and 1 are enhancer and inhibitor respectively, 4 is the product protein and will be translated into logic circuit gate through a modulo function (see section C). The last two integers are required to define the input connections to the logic gates and are similar in structure to the connections used in Cartesian Genetic Programming (CGP) [4]. These connections are automatically adjusted during gene regulations.

0	1	4	0, 2

Figure 1: One Gene

B. Development

In our model, as in biology, the transcription process of protein synthesis produces mRNAs that are released from gene regulations in cells. This releases differential constructing functions in cells when gene regulation patterns are different. Multi cellularity is obtained from cell division processes. It is determined by predefined division proteins and occurs during gene regulation. Cell division involves two steps: copying a mother cell's proteins; and obtaining the final cell proteins through a protein signalling pathway (see later). Differential gene regulations in cells result from cell signaling, cell division and potentially different gene regulation loops. Cell signaling includes protein diffusion through a cell membrane threshold and protein alteration as a consequence of a signaling pathway [7, 8]. Figure 2 gives an example of signaling processes. It shows the signaling associated with a particular protein (4 in this case). When the level of diffusing protein is larger than the membrane threshold 30.0 it is able to diffuse and change to protein 1 in its neighbor cells. The level of protein to diffuse is uniform and calculated from equation 1 as the diffusion in [5].

Protein 4 membrane threshold: 30.0 Pathway: 4→1

Figure 2: Cell Signaling

$$P^{t+1}_{conc}=1/2P^{t}_{conc}+ 1/(2*NumberofCellBorders)\sum P^{t}_{conc} \quad (1)$$

C. Circuit Construction

A circuit is constructed during gene regulation and development by a process that is inspired by the protein synthesis process in biology. Figure 3 gives an overview of the construction process compared with the protein synthesis process in biology for a 3 gene sequence. In biology, proteins are constructed by amino acids which are generated from gene regulation and mRNA translation. In the electronic circuit design, elements required for design such as gates (for the gate-level design) are generated from gene regulation, products

S. Zhan is with the Department of Electronics, University of York, UK. (sz503@york.ac.uk)

J. F. Miller is with the Department of Electronics, University of York, UK. (jfm@ohm.york.ac.uk)

A. M. Tyrrell is with the Department of Electronics, University of York, UK. (amt@ohm.york.ac.uk)

978-1-4244-2959-2/09/$25.00 ©

generated as in figure 3: 6 4 1 0 6 2 and translated to gates by the mathematical modulo function as in table 1. After this the gate identifiers become (figure 3): 2 0 1 0 2 2 (NAND, XOR, OR, XOR, NAND, NAND). The gates are connected together by a CGP netlist which is analogous to the process of linking amino acids to construct polypeptide chains through a sequence of gene regulations. The constructed circuits build cell functions as illustrated in figure 4 and these circuit cells are connected together to obtain the overall cellular system function using another CGP netlist as illustrated in figure 5. Table 1 is the modulo translation look up table with 8 types of regulatory product proteins (0 – 7) that is used with the example shown in figure 3.

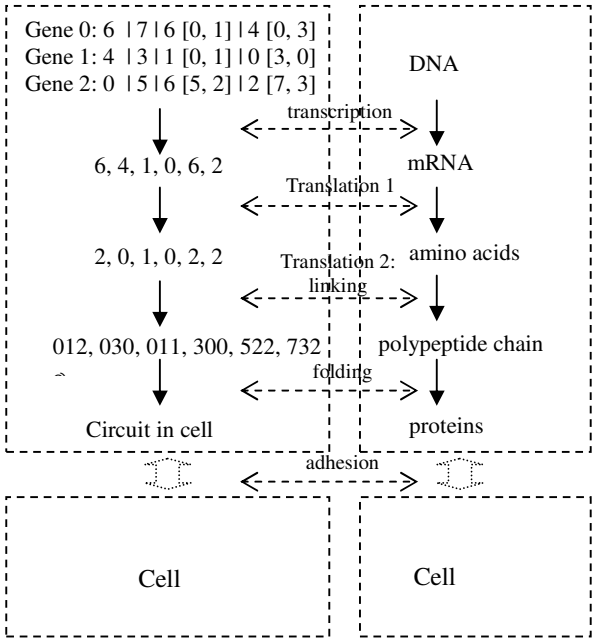
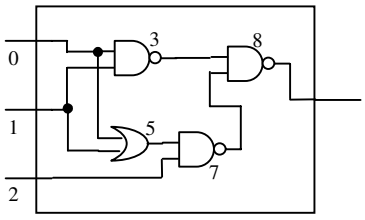


Figure 3 circuit construction processes



netlist: 012, 030, 011, 300, 522, 732

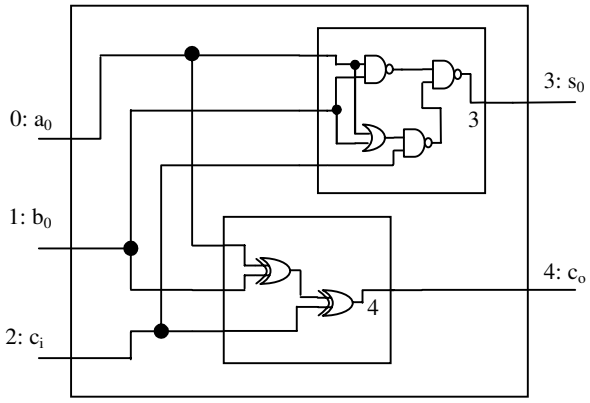
Figure 4: Constructed circuit in one cell

D. Evolution

A 1+4 evolutionary algorithm is used to search genes, gate connections, products, production rate (used to determine the level of each product), membrane, pathway, zygote proteins and cell connections. Figure 6 shows one example of each element evolved in the case of evolving a 1 bit full adder. In this example, 5 genes are used, each with one enhancer and one inhibitor and two regulation products. Each product is associated with connection as seen in the square bracket. Gene Product production rates are defined by a floating point numbers. These can be seen at the end of each gene. 12 proteins are available for the development and therefore, 12 elements are associated with cell pathway and cell membrane (0 – 11). The initial evolved protein in the zygote is protein 9 which initializes the gene regulations. Individuals are developed in a two cell environment. There are three inputs for each cell and therefore 6 connection labels (see the cell connection section in figure 6).

Table 1 Gene Product-Gate translation (mod 4)

Gate Id	Product protein Id
0 (xor)	0, 4
1 (or)	1, 5
2 (nand)	2, 6
3 (nor)	3, 7



netlist: 012, 012, 34

Figure 5: Cell Connection

/----------Genome----------/
Gene 0 : 0 | 5 | 9 [1, 0] | 5 [0, 1] | 0.90
Gene 1 : 7 | 0 | 2 [4, 3] | 6 [1, 0] | 0.11
Gene 2 : 2 | 3 | 10 [4, 4] | 7 [7, 2] | 0.52
Gene 3 : 6 | 0 | 0 [1, 3] | 0 [7, 6] | 0.10
Gene 4 : 9 | 10 | 2 [8,10] | 9 [6, 2] | 0.05
/----------Cell Pathway ----------/
0 | 4 , 1 | 2 , 2 | 9 , 3 | 8 , 4 | 11 , 5 | 10 , 6 | 2 , 7 | 6 , 8 | 9 , 9 | 10 , 10 | 0 , 11 | 1
/----------Cell Membrane ----------/
0 | 100.0 , 1 | 30.0 , 2 | 90.0 , 3 | 60.0 , 4 | 40.0 , 5 | 30.0 , 6 | 10.0 , 7 | 10.0 , 8 | 30.0 , 9 | 60.0 , 10 | 80.0 , 11 | 100.0
/----------Initial Zygotes ----------/
9
/----------Cell Connection ----------/
2 01 0 01

key:
Gene Id: Enhancer | Inhibitor | Product 1 [Connection 1] | Product 2 [Connection 2]| production rate
Pathway: protein id | transferred protein id
Membrane: protein id | threshold

Figure 6: Example of Evolved Individual

III. Natural Mechanisms

A. Self-Organization

With gene regulation and cellular development, organisms maintain themselves in particular steady states. Changes of environmental parameters that drive the systems to other states, can be viewed as the system's adaptivity to the environments. For instance, a chameleon changing its skin colour to match with the environment and avoid enemies. When the system develops into a fixed state, changes in the system such as missing cells or changing proteins can cause faults in the systems. The ability to reconstruct themselves into either the original state or an alternative state prepares the system to work in the changing environment. From the design point of view, adaptivity and self-reconstruction are desirable characteristics for systems working in noisy and faulty environments.

In [8] we investigated a typical evolved individual's response to cell removal and protein damage. The paper showed that the individual has the ability to recover to the original state after damages but, in some situations will also move to other states. These new steady states alter depending on the pattern and level of damage. Figure 7 shows the change of state of the same individual developing in an 8 cell environment after a series of random cell removals and protein damage at its stable regulation states. In this case, the gene regulations of the organism return it to its original state. In each state from 1 to 10, the top figure shows the cell types (determined by the largest protein). This only shows us that there is damage in the individual and we are interested in what happens to the gene regulation states. These are shown in the sequence of corresponding figures under the cell state. Damage to the system starts after state 1, with the removal of six cells the individual phenotype comes to state 2. Gene regulations arrive at the regulations shown underneath (number 2) at state 2. After a series of randomly imposed protein changes (states 3-7) and corresponding gene regulations, eventually it runs into states 8 and state 9. Finally, changing proteins at this state causes the cells to return the individual to gene regulation state 10 which is the original gene regulation state 1.

B. Evolution in Damaging Environments

According to natural evolution theory, the environment plays a major role in the evolution of species and in order to survive evolved species build up an adaptivity to environmental changes and robustness. Thus we would expect that if the agents are evolved in a changing environment eventually the agents will be more adaptable to similarly changing environments and also be robust.

To obtain individuals that are robust to system damages, random damages could be applied during evolution as described in section IV.

C. Degeneracy

Edlman et al. in [2] argue that degeneracy (the ability of elements that are structurally different but perform the same function) is both necessary for, and an inevitable outcome of natural selection. It is common at all levels of organization in biological systems such as genes, neural networks and immune systems [2] but not common in man-made systems. With this property, the system should perform better than systems with redundancy alone in terms of evolution, as redundant systems require copies of structurally identical systems but evolution operations tend to alter the system's elements [1].

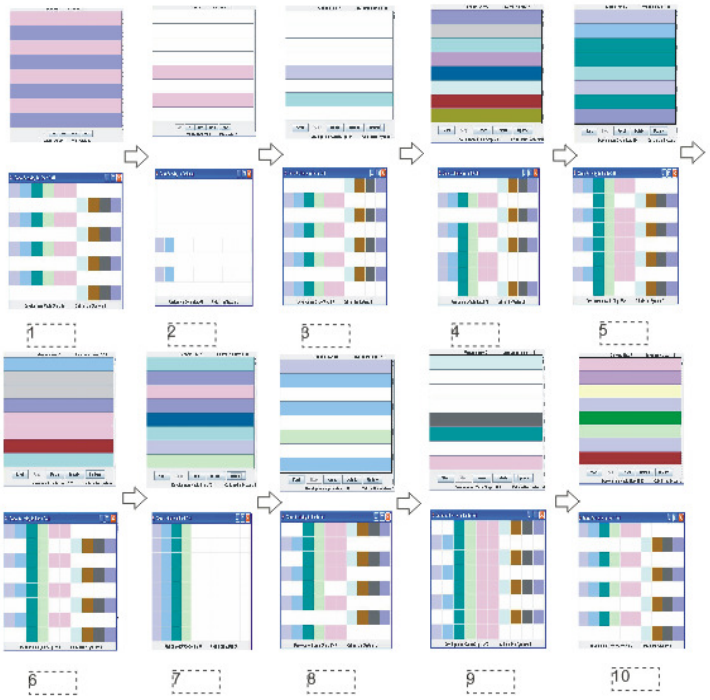


Figure 7 State Alteration after damage

During mathematical modulo gene product proteins and gate translation, one gate function can correspond to many gene products, this mimics mRNA codon to animo acid translation in protein synthesis and provides degeneracy in the modeled genetic code. In table 1 each gate can be produced from two gene products (we call this gate level degeneracy factor 2 in the later experiments). Figure 8 gives an example of a redundant functions provided from two degenerate genes.

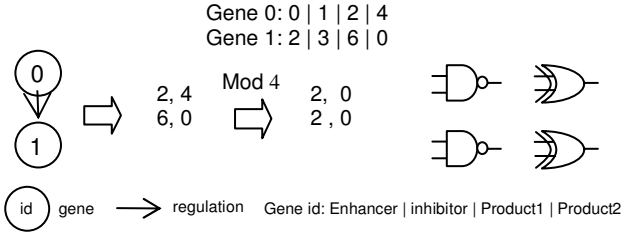


Figure 8 Genetic Code Degeneracy

Genes with different regulatory sites but the same products will generate functional redundant gates as in the example in figure 9. The same translation function and gate types are provided as in figure 8. Genes once regulated will generate the same products which will be translated to the same gates.

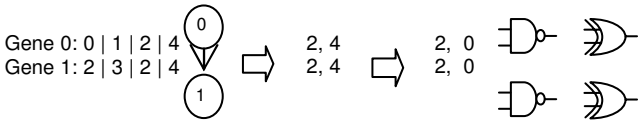


Figure 9 Gene Regulatory Site Degeneracy

Different environmental contexts can cause different gene regulations that generate the same structures and present degeneracy during regulation. Figure 10 shows an example of two different genes. They regulate independently with different context stimulation and generate the same products for the same gates.

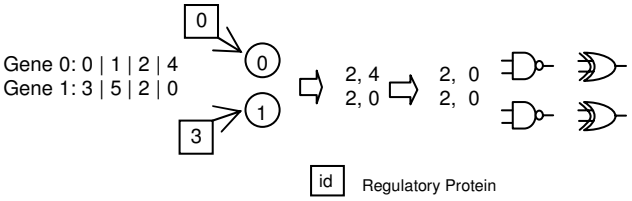


Figure 10: Regulation (Slicing) Degeneracy

When the combination of regulatory proteins is determined by the gate types, the same gene can be regulated by different regulatory proteins. This means that there is regulatory protein degeneracy. Figure 11 shows one example of this, where both regulatory protein 0 and 4 after mod 4 translation can bind with regulatory site 0 and therefore regulate gene 0's action and generate gates.

This type of regulatory site binding with mutation operations will introduce gene neutrality during regulations as the example shown in figure 12. The existence of neutral genes is underlined. This is obtained from allowing a larger mutational range to gene regulatory sites. In this example, the regulatory binding range is 0, 1, 2, and 3 but twelve ids are involved in evolution. Gene 1 and 2's enhancer ids are both larger than 3 so they will not be involved in gene regulations and will always be neutral genes. The larger the degeneracy level, the more opportunity there is for neutral genes to exist during regulation in the system.

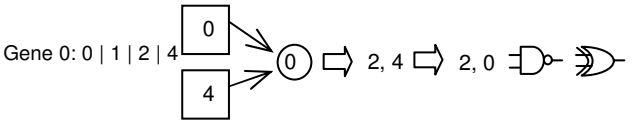


Figure 11: Regulatory Protein Degeneracy

Gene 0 : 0 | 4 | 3 | 8 | 1 2 | 2 1 | 0.90
Gene 1 : 4 | 5 | 8 | 7 | 3 1 | 4 0 | 0.78
Gene 2 : 5 | 8 | 11 | 7 | 3 3 | 2 2 | 0.65
Gene 3 : 2 | 5 | 8 | 7 | 4 3 | 9 4 | 0.09
Gene 4 : 0 | 0 | 9 | 11 | 9 7 | 5 10 | 0.84

Note:
Gene Id: Enhancer | Inhibitor | Product 1 | Product 2 | [Connection 1] | [Connection 2] | rate

Figure 12: the example of presented neutrality

Incorporating the same gate connections in degenerate genes or gene regulations will construct degenerate circuits as the example in figure 13 where gate 3 and gate 5 generate the same outputs from different gene actions.

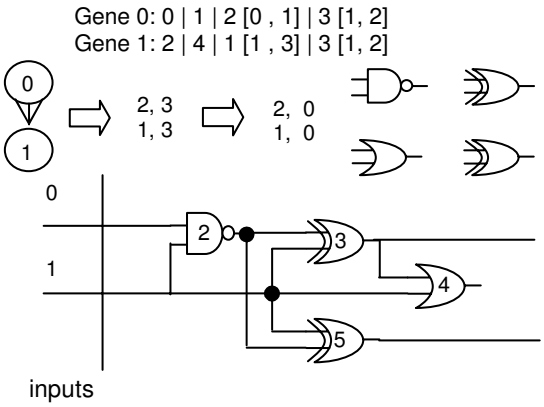


Figure 13: Circuit Construction Degeneracy

Using a CGP representation approach allows degeneracy since different netlists will construct to same functions [4]. Figure 14 shows two different translated netlist that yield the same output from gate 6. The first and third translated have no affect on this output.

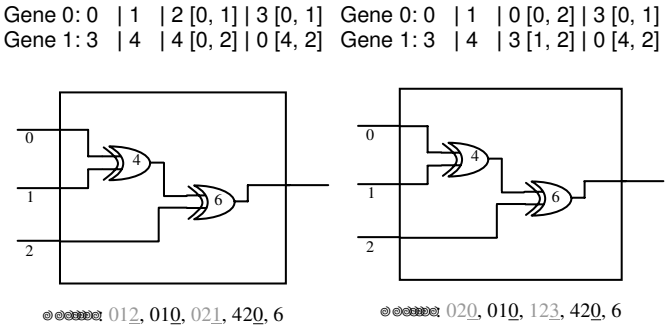


Figure 14: Degeneracy from Neutral Gates

IV. EXPERIMENTS

We have investigated our model in the context of evolving some small digital circuits: 1 bit adder, 2 bit multiplier and 3 even parity in environments that are either not damaging or there is a constant level of damage. It is important to note that our aim is to investigate the

characteristics of the circuits not to evolve simply those simple circuits.

We carried out two types of experiments. In one, individuals were evolved in a damaging environment so that they will be able to continue to function when environmental damage is present. The first set of experiments were carried out to investigate how evolution obtains robustness in damaging environment. In the second experiment different amounts of degeneracy are introduced to investigate its affect on evolutionary search to find robust solutions.

Since circuit functions are obtained from individuals' developmental process, damage during development is considered as serious damage and investigated in the experiments.

A. *Gene Knockout Damage*

We have investigated gene knockout. This is very serious damage as we expect the remaining undamaged parts of the system to carry out the complete function. In biology, gene knockout is a technique that is used to learn about gene functions but it also demonstrates degeneracy. Figure 15 gives an example of gene knockout damage. It represents a 5 gene individual set into a 3*3 environment. 1 gene is knocked out in each cell, e.g. in the top left corner cell, gene 1 is knocked out and in the cell next to it, gene 0 is knocked out, ans so on for each cell.

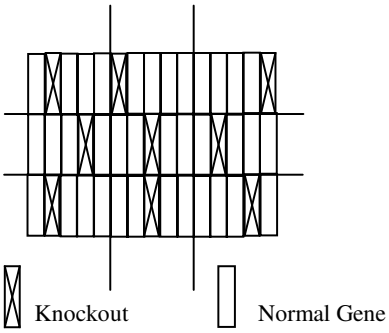


Figure 15: one knockout example

B. *Protein Alteration Damage*

Protein alteration is a form of transient damage, however, in a regulatory dependent process, it often has a tremendous effect on the development process. In these experiments, the protein Id is randomly altered in each cell during development. Figure 16 gives an example of protein id being altered. The left figure shows proteins in the cell before damage and the right after half the protein id have been changed.

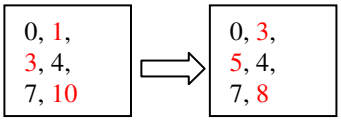


Figure 16: one protein id alteration example

V. RESULTS AND ANALYSIS

A. *Robustness Measurement*

We can evaluate the maintenance of evolved functions when running the individuals in a changing environment using a measure of robustness. We have defined this measure so that it has a value between 0 to 1. This can be obtained by dividing the current function fitness (after damage) by the evolved function fitness. The higher the ratio the more robust the system. The overall effect of different damage can be assessed through the average of individuals' robustness in random damage. This involves two steps: firstly to assess the average robustness of each individual in many random damage and next to obtain the average robustness of these individuals.

B. *Parameters*

Table 2 lists the parameters used during evolution and development. Initially 10 individuals are randomly generated to provide the possibility of population diversity. The best individual is chosen as the parent in each generation. This parent generates 4 children through mutations of the current generation. Mutation operations randomly mutate elements in each generation. Selection and generating children processes occur at every generation and stop at a predefined evolution generation: 20000. In the case that no better individuals are generated, but a child has a fitness equal to the parent's the child will be selected to allow some genetic drift [4]. Development starts from one single cell at predefined location (0, 0) with an evolved initial protein with concentration 100. Fitness will be evaluated from the tenth development cycle and will be continually evaluated for 10 development cycles. This was done in order to allow stable gene regulations to be arrived at. Individual fitness is the average fitness at each development cycle based on their truth table. Each set of results is averaged from 30 individual runs.

Tables 3 to 5 list the experiment parameters and the damage parameters during and after evolution. Each function is evolved with its own set of parameters: input number, cell input number, and circuit output number and sets of logic gates involved in evolution and environment size for their development considering their domain knowledge. Degeneracy factors are set from 1 to 6 for each function associated with different number of proteins presented in table 4. During development, damage is applied throughout the development cycles from cycle 0 to cycle 19. One gene randomly chosen in each cell is knocked out in the gene knockout experiments and in the protein alteration experiments half of the proteins in each cell are randomly changed to others.

C. *Results*

Evolution of system functionality

Figure 17 shows the average best fitness of 30 runs of each function. Each figure presents three sets of evolutionary experiments: evolution only (no damage),

gene knockout damage during evolution and development, and finally, evolution with protein alteration damage during evolution and development.

Table 2: Evolution Search Parameters	
Initial random individuals	10
Start fitness evaluation at development cycle	10
Continue fitness evaluation cycles	10
Mutation rate	0.025
Evolution Generations	20000
Evolution run number	30
development parameters	
Gene Number	5
Number of infused zygote proteins	1
Zygote cell position	(0,0)
Zygote Protein Concentration	100

Table 3: Circuit Construction Parameters				
functions	Circuit input No.	Circuit cell inputs No.	Circuit output No.	Gate Set (id)
1-bit adder	3	3	2	Xor (0) And (1) Or (2)
2-bit multiplier	4	4	4	Xor (0) And (1)
Even-3 parity	3	3	1	And (0) Or (1) Nand (2) Nor (3)

Table 4: Development Parameters						
functions	Number of Proteins					
	Degeneracy level					
	1	2	3	4	5	6
1 bit adder	3	6	9	12	15	18
2 bit multiplier	2	4	6	8	10	12
Even 3 parity	4	8	12	16	20	24

Table 5: Damage parameters	
Gene knockout rate	0.2 (1 gene)
Protein alteration rate	0.5
Damage Cycles	0-19

Comparing the second column with the first in each set in figure 17, it can be seen that gene knockout damages in many cases actually assists evolution's search to functions. This partly demonstrates the contribution of degeneracy in evolution because if the damaged systems still carry the complete functions, the knockout genes must either contribute to redundant function or be neutral genes.

In figure 17 we can see that degeneracy is often helpful to evolution as it causes high average fitness to be obtained. It is especially helpful in situations where damaging environments occur during evolution. In some cases evolution with gene-knockout and some level of degeneracy obtains better fitness than evolution in undamaging environments and low levels of degeneracy (see the one-bit adder case in Fig 17, in particular). Degeneracy is particularly helpful to evolution in protein damaging environments. As discussed in [2] degeneracy makes major contribution to biological complexity, therefore we think from the evolution point of view, the ideal evolutionary process must find the trade off between the variability and the complexity provided by degeneracy.

Degeneracy impact on Robustness in non-damaging environments

The way robustness is influenced by degeneracy in genetic code is seen in figure 19.

The first column in each set in figure 19 shows the average robustness of the 30 best individuals evolved with no damage during evolution to random gene knock-out and protein alteration with various levels of degeneracy.

Most of the graphs clearly show that genetic code level degeneracy does have a visible effect on the robustness even though these individuals are obtained from pure evolution (without damage being introduced during evolution).

Inherited in the different level of gene code level degeneracy, the introduced neutrality during gene regulations is the important aspect to the contribution of robustness. When we change the mutation range on the regulatory sites and product sites of genes, we can remove the neutrality introduced by degeneracy. Figure 18 shows how robustness to gene knock out damage is affected when neutrality is either present or not. It can be seen that the presence of neutrality improves the robustness in each case.

Robustness of evolved solutions in damaging environments

The second column in each figure of figure 19 shows the average robustness of the evolved best individuals from 30 independent evolutionary runs in damaging environments involving random gene knock-out at the left of figure 19 and protein alteration at the right of figure 19.

Since the damage is present during evolution, evolution has an extra hidden objective, to search for individuals which are robust. It reveals again the effect of degeneracy. Also that the evolutionary process is influenced by the variability and complexity provided by degeneracy and tries to achieve the best trade off between highest degeneracy (level 6) and no degeneracy (level 1).

Evolved solutions are more robust when evolved in damaging environments.

In figure 19 the robustness obtained from pure evolution are compared with the ones from evolution in damaging environments. Each comparison shows there is a clear improvement in the robustness in solutions found by evolution in damaging environments.

VI. CONCLUSIONS

Robustness is a necessary characteristic for a system to function in damaging environments. Looking beyond the traditional robustness techniques, this paper investigates some biological inspired processes: evolution in damaging environments, degeneracy and neutrality during gene regulations. We find that robustness arises naturally from these mechanisms.

Processes such as evolution with gene knockout damage also contributes to the evolution search process to obtain better fitness sometimes. Degeneracy also affects evolutionary search especially in damaging environment and the best search process needs to find the trade off between the variability and the complexity it provides to the system.

It is believed that these characteristics exist in every scaled circuits with this design method. Degeneracy is inherited from the underlying design mechanism and evolution operators operate at the genotype level. Future work on this issue will move on to evolve larger circuits and investigate their characteristics.

REFERENCES

[1] Fernandez, P. and Sole, R. : The role of computation in complex regulatory networks. In Koonin, E. V., Wolf, Y. I., and Karev, G. P., editors, Power Laws, Scale-Free Networks and Genome Biology. Landes Bioscience, 2004

[2] Gerald M. Edlman and Joseph A. Gally,: Degeneracy and complexity in biological systems, PNAS, vol. 98, no. 24, 13763-13768, 2001

[3] G Gershenson, C., S. A. Kauffman, and I. Shmulevich:. The Role of Redundancy in the Robustness of Random Boolean Networks. Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems. pp. 35-42. MIT Press, 2006.

[4] Miller, J. F, Thomson, P. : Cartesian Genetic Programming, Third European Conference on Genetic Programming, Proceedings published as Lecture Notes in Computer Science, Vol. 1802, pp. 121-132, 2000

[5] Miller, J. F. : evolving a self-repairing, self-regulating, French flag organism, Proceedings of GECCO, Part I. LNCS, 3102 Springer 2004 pp. 129-139.

[6] Wagner, A. (2005). Robustness and Evolvability in Living Systems. Princeton University Press, Princeton, NJ.

[7] Zhan, S, Miller, J. F., Tyrrell, A. M. : An Evolutionary System using Development and Artificial Genetic Regulatory Networks, IEEE Congress on Evolutionary Computation (CEC), p812-822, 2008

[8] Zhan, S. Miller, J. F, Tyrrell. A. M. : A Developmental Gene Regulation Network for Constructing Circuits, International Conference on Evolvable Systems: From Biology to Hardware (ICES), pp177-188, 2008

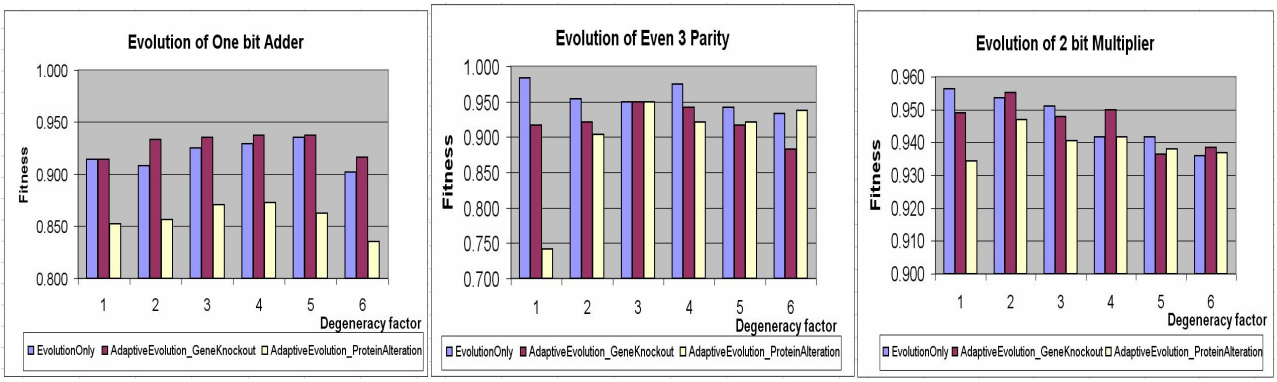


Figure 17: Average fitness (30 runs) on three circuit problems evolved in a non- damaging environment compared with evolution with gene knockout and protein alteration and its change with various levels of degeneracy

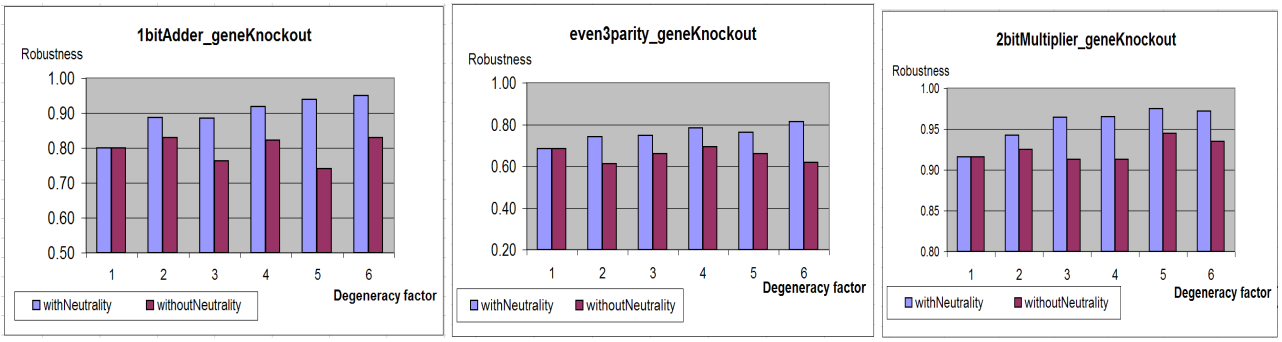


Figure18: Average robustness (30 runs) comparison of with neutrality and without it on three circuit problems evolved in non damaging environment for gene knockout damage with various levels of degeneracy

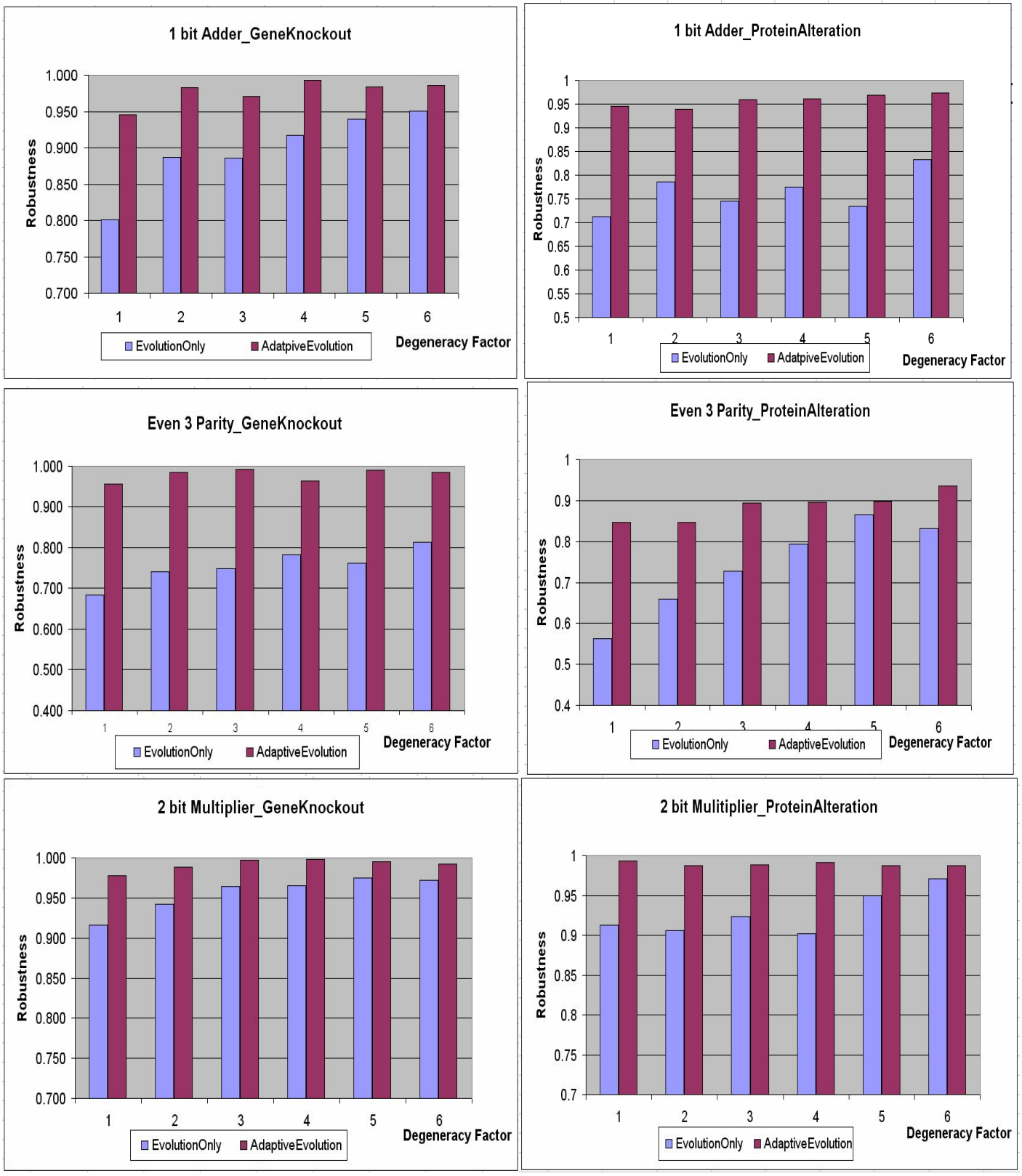


Figure 19: Robustness: the contribution of evolution in damaging environment (AdaptiveEvolution) compared with evolution in non-damaging environment with various levels of degeneracy. The average fitness of each column before the evolved individuals are set into damaging environments is presented in figure 17.