2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

An Examination of Synchronisation in Artificial Gene Regulatory Networks

Jonathan Byrne, Miguel Nicolau, Anthony Brabazon and Michael O'Neill
Natural Computing Research & Applications Group,
Complex and Adaptive Systems Lab, University College Dublin

Abstract— **An Artificial Genetic Regulatory Network (GRN) is a model of the gene expression regulation mechanism in biological organisms. It is a dynamical system that is capable of mimicking non-linear time series. The GRN was adapted to allow for input and output so that the system's rich dynamics could be used for dynamic problem solving. In order for the GRN to be embedded in the environment, the time scale of the physical system has to be mapped to that of the GRN and so a synchronisation process was introduced. This work examines the impact of different synchronisation intervals and how they effect the overall performance of the GRN. A variable synchronisation step that stops once the system has stabilised is also explored as a mechanism for automatically choosing the interval size.**

I. Introduction

The understanding of biological mechanisms involved in the evolutionary process has advanced significantly in recent years. Accordingly, the models of evolutionary computation have changed to incorporate this knowledge. A gene in a cell is not always expressed, instead gene expression is dependent on the complex interaction of proteins and the other regulatory mechanisms in the cell.

Artificial Genetic Regulatory Networks (GRN) were initially developed to understand the behaviour found in biological regulatory networks that adaptively control the expression of RNA and proteins in a cell [3]. Such techniques gave insight into phenomena such as heterochrony, a variation in timing expression. The ability to react dynamically to changes in the environment led to the development of GRNs into a machine learning technique for dynamic environments. The GRN model has since been applied to dynamic problems, such as balancing a pole [4] or modelling financial data [5].

The experiments described in this work are based upon the GRN model devised by Banzhaf [1]. In order for the model to handle inputs and generate outputs, a number of changes were made to the original Banzhaf model. A synchronisation step was added which allowed the system to perform several iterations after a change to the input, thus giving it time for the system to stabilise after perturbation. This work investigates the effect of the synchronisation step on the performance of the algorithms on two different problem domains.

The results show that a many to one synchronisation mapping provides a clear benefit, but that efficacy is reduced beyond a certain number of iterations. A one to one mapping, where an iteration of the physical system is mapped to a single GRN iteration, performs poorly and in some cases fails to find a solution. Even increasing a physical iteration to ten GRN iterations provides a clear benefit. Increasing this value further improves the performance of the system but this improvement scales logarithmically as the number of iterations is increased.

This paper is organised as follows. Section II gives a description of the GRN model and explains how it is composed of evolutionary and developmental processes. Section III gives an overview of the synchronisation step and describes the different settings used in this work. Section IV describes the evolutionary algorithm used and states what experimental settings were used for the experiments. Section V details the effect of changing the synchronisation step for the pole-balancing experiments. Section VI examines how the sync step effects the GRN's ability match an input with an offset output. Finally the results and future work are discussed in Section VII.

II. The GRN Model

GRNs are composed of a network of biological interactions between the genes in a chromosome and the proteins they produce: each gene encodes specific types of protein. Certain proteins termed Transcription Factors regulate (either enhance or inhibit) the expression of other genes which, in turn, effect the generation of the protein those genes encode.

There are two elements to the GRN model used in this work. They represent the combination of evolutionary and developmental processes that occur in biology. The first component is a binary representation which is evolved to optimise the GRN for a particular problem. The second component is generated from the binary representation by a mapping process that generates a recurrent network. Both of these components are described in detail in this section.

A. The Evolutionary Representation

The evolutionary component is used to "tune" the model for a particular problem. Initially random GRNs are generated and tested on the problem domain. The GRNs with better performance are preferentially selected and used to create the next generation. This process iteratively tunes the GRN on the specific problem domain.

The representation is composed of a genome, represented as a binary string. The proteins interact with the genome at regulatory sites in the gene. The genome is scanned for promoter sites, identified by a specific binary sequence. If such a site is found this indicates the presence of a gene. The binary signature consists of an arbitrarily selected 32 bit pattern: the sequence XYZ01010101 identifies a gene, with X, Y and Z each representing an arbitrary sequence of 8 bits. If a promoter site is found, the 160 bits (5×32) following it represent the gene sequence, which encodes a protein.

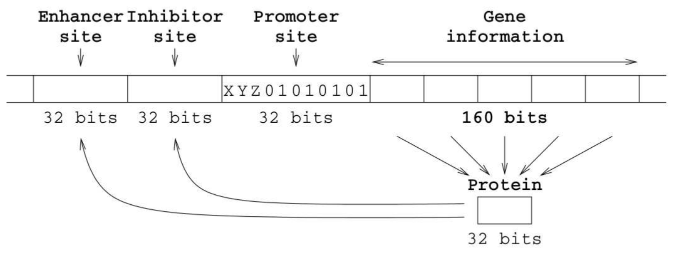

Fig. 1. Bit string encoding of a gene. If a promoter site is found, the gene information is used to create a protein, whose quantity is regulated by the attachment of proteins to the enhancer and inhibitor sites.

The gene sequence is mapped from a 160 bit sequence to a 32 bit protein sequence. The 160 bits are split into five 32 bit sets. The majority rule is used to decide the value of a bit by observing the value in the sets at that position. The effect that a protein has on other proteins in the network is defined by the 64 bits upstream from the promoter site. Each 32 bit segment represents the enhancer and inhibitor sites for a gene. Figure 1 illustrates the encoding of a gene. An XOR operation is used to calculate how well the proteins bind with the regulatory site. The XOR operation returns the number of complimentary bits between both string. The enhancing and inhibiting signals regulating the production of the protein are calculated by the following equation:

$$e_i, h_i = \frac{1}{N}\sum_{j=1}^{N} c_j \exp(\beta(u_j - u_{max})) \quad (1)$$

where N is the total number of proteins, c_j is the concentration of protein j, u_j is the number of complementary bits between the (enhancing or inhibitory) regulating site and protein j, u_{max} is the maximum match observed in the current genome, and β is a positive scaling factor. Because of the exponential, only proteins whose match is close to u_{max} will have an influence here.

B. The Developmental Model

Once the genome has been mapped it defines a network of genes, connected by their enhancing or inhibiting effects on each other. The concentration of the protein for a particular gene depends on concentration of other proteins and whether they have an enhancing or inhibiting effect on that gene. The production of p_i is calculated via the following differential equation:

$$\frac{dc_i}{dt} = \delta(e_i - h_i)c_i - \Phi(1.0) \quad (2)$$

where δ is a positive scaling factor (representing a time unit), and $\Phi(1.0)$ is a term that proportionally scales protein production, ensuring that $\sum_i c_i = 1.0$.

The original Banzhaf model only had one type of gene in the model, the transcription factor gene, as shown in Figure 2. In order to handle input and output the genes were divided into two groups, Product (P) genes and Transcription Factor (TF) genes. The concentration of each P protein is read as output and they are affected by the concentration of TF proteins. The concentration of P proteins is normalised separately, otherwise the combined concentrations of all the proteins in the network would suppress the changes to the P protein concentrations. TF genes are akin to the traditional Banzhaf model where every TF gene effects every other TF gene. The input mechanism is provides by EXTRA proteins which are essentially TF proteins where the input value sets the concentration. Changing the concentration of EXTRA proteins allows the input to effect the concentration of the other TF proteins in the network. An illustration of a typical network is shown in Figure 3.

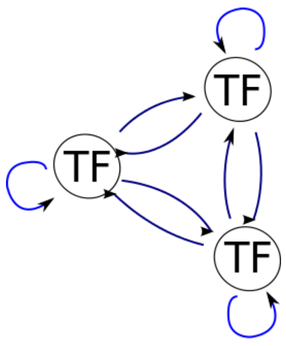

Fig. 2. The original Banzhaf model.

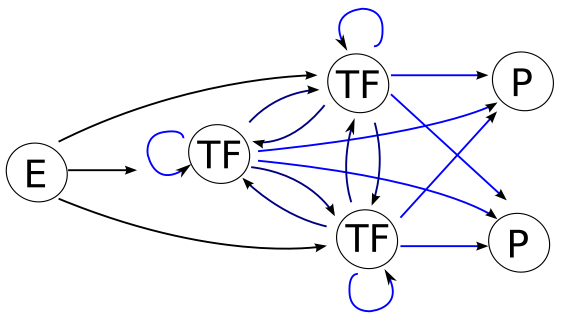

Fig. 3. The input is provided by the concentration of the E proteins and affects all TF genes. The TF proteins in turn affect the concentration of the P proteins.

III. The Synchronisation Step

In order to allow the GRN model to interact with a physical system, a synchronisation process is required to map the GRN iterations to that of the physical system. The original pole balancing work had an interval of 2000 time steps as the GRN performed 1000 iterations in 0.01 seconds and the physical model for the pole balancing experiment updated every 0.02 seconds [4]. The size of the synchronisation step effects the dynamics of the GRN model as it allows the model to respond to input changes by allowing it to iterate for a fixed number of steps. The aim of this work is to examine how tuning this parameter effects the performance of the model. The fixed intervals examined are 1, 10, 100, 500 and 1000. A variable synchronisation process is also explored where the synchronisation period will end if the total change, Δ, in concentrations between iterations drops below a certain level. This is expressed mathematically as:

$$\Delta = \sum_{P=1}^{Pn} |P_t - P_{t-1}|$$

where P_t is the current concentration of the protein and P_{t-1} is the concentration from the previous iteration. This provides an intriguing possibility of a self tuning parameter that would choose the right synchronisation period for different problem domains. Another advantage of stopping when the system has reached a stable state is that unnecessary iterations are not performed. The criteria for a stabilised system is that the sum of the absolute value of all changes in an iteration are less than 1×10^{-5}. If the total change did not drop below this level after 1000 iterations, then the synchronisation step was stopped.

IV. Experimental Setup

The GRN is optimised using an evolutionary strategy [6] and applied to three different problems, pole balancing, offset sine and offset random walk. The evolutionary algorithm used to evolve the binary genomes is an evolutionary strategy $(250 + 250) - ES$: 250 parents generate 250 offspring, and the best 250 of all 500 are used as the new parent population; a maximum of 50 iterations were allowed. The only variation operator used is a simple bit-flip mutation, set to 1% and changed during the course of the run according to the $1/5$ rule of Evolution Strategies [6]: when the rate of successful mutations is higher than $1/5$ (i.e. when more than 20% mutation events result in improved fitness), the mutation rate is doubled; it is halved in the opposite case. However, to avoid stagnation of evolution, if the number of mutation events (i.e. the number of bits flipped per generation) drop below 250, the mutation rate is doubled. Thirty runs were carried out for each experiment.

V. Pole Balancing Experiment

The pole-balancing problem, also known as the inverted pendulum problem, is a benchmark for dynamic experiments [2, 7]. The task consists of controlling a cart along a finite one dimensional track. There is a pole fixed to the cart and the objective is that the pole must remain balanced on the cart while the cart remains within the boundaries of the track.

The problem is described with four variables:

$x \in [-2.4, 2.4]$ m is the position of the cart from the centre;

$\theta \in [-12, 12]$ $^\circ$ is the angle of the pole with the vertical;

$\dot{x} \in [-1, 1]$ m/s is the velocity of the cart on the track;

$\dot{\theta} \in [-1.5, 1.5]$ $^\circ/s$ is the angular velocity of the pole.

The physical model updates every $0.02s$ during the simulation. The experiment is carried out for 120,000 time steps or until either the cart reaches the track boundaries ($x = \pm 2.4m$), or the pole falls (i.e., $|\theta| > 12^\circ$). The GRN requires four EXTRA proteins to handle the four inputs and outputs one of two possible values, push the cart left or right (with a constant force $F(t) = \pm 10N$).

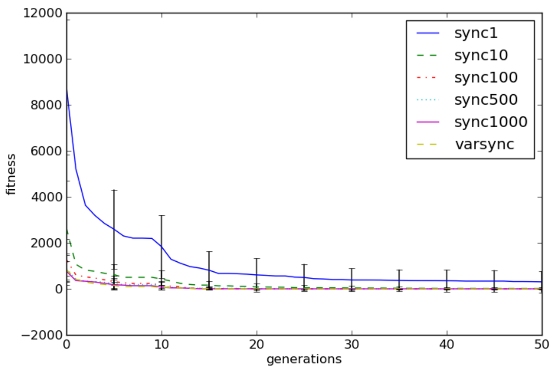

Fig. 4. Results for the Pole balance experiment over thirty runs. The error bars show the variance of the runs.

As the GRN can start in a highly dynamic state so there is an initial stabilisation step where the GRN is allowed to settle, it is then tested against a random cart state. The cart position changes for every generation and so the fitness function is very noisy. The random initialisation can also be problematic as several combinations of the four input variables result in unsolvable states (i.e. the pole cannot be balanced). The fitness is calculated using the following equation:

$$F(x) = \frac{120000}{\text{successful time steps}}$$

A time step is considered successful if the cart does not exit the $\pm 2.4m$ track, or the pole does not fall beyond the $\pm 12^\circ$ range. Output is read from a single P protein, if the concentration is greater than 0.5 then the cart is pushed to the right, if it is less than 0.5 it is pushed to the left. As the number of P genes is arbitrarily defined by the genome, the concentration of each P protein is tested and the best result is assigned to the GRN.

A. *Pole Balance Results*

The results in Figure 4 show that the synchronisation step improves the performance of the GRN although the improvement scales logarithmically for greater step sizes. The results for the one to one mapping, that is a step size of 1, improve up to generation thirty but eventually plateau and do not reach the same level as the other step sizes. The variable step size operator performs as well as the step sizes larger than one.

The evolutionary algorithm is stopped as soon as it managed to find a solution (120,000 time steps). In order to investigate whether the generated solutions were robust a generalisation test is carried out [7]. The generalisation test varies the four input variables, with their normalised values set to the following: 0.05, 0.275, 0.50, 0.725, and 0.95, resulting in $5^4 = 625$ test cases. The generalisation score of the best individual found is thus the number of test cases out of these 625, for which the controller manages to balance the pole for 1000 time steps.

Sync Step	Successful Runs	Ave	Std
Sync 1	5	22.63	53.96
Sync 10	26	182.7	109.49
Sync 100	29	243.6	128.49
Sync 500	29	171.03	136.64
Sync 1000	29	200.93	133.06
Variable	29	207.9	120.5

TABLE I
GENERALISATION RESULTS FOR THE SUCCESSFUL RUNS.

The generalisation results for GRN in Table I highlight the benefit of having a synchronisation step. Step size one only generated five solutions that could balance a pole whereas greater step sizes generated many more solutions. In order to compare the results, runs that did not generate a successful solution were given a generalisation score of zero. Upon further investigation it was found that solving the problem early in the run generated solutions that generalised poorly, as it resulted from an easily solved initial configuration. As increasing the synchronisation step improved convergence this turned an advantage into a disadvantage as quicker convergence created runs with poor generalisation scores. Despite this, the average generalisation scores for high synchronisation steps were equivalent.

VI. Offset Experiments

Two experiments are conducted in this section that examine how increasing the delay interval between the input and the desired output effects the performance of the GRN for different synchronisation steps. The sine and random walk experiments initially examine if a GRN is capable of replicating a given input as an output. The desired output is then offset to investigate if previous input, in conjunction with a synchronisation step, improves convergence on the desired output. The difference between both experiments is that the sine wave has a regularity to the input whereas the random walk experiments have no underlying pattern. There is also another change in the experimental settings from the previous experiment, the number of generations is increased to 250 to observe if there is any additional improvement over time.

In order to properly understand the results and their effects on performance, they were compared with two methods. The first baseline is copying the input. If all it is doing is redirecting the input to output then the model is not truly reacting to the environment but merely re-expressing it. the second baseline is a simple deterministic method, linear regression. This deterministic method provides a baseline for performance as the algorithm should outperform fitting a line to the data.

A. Sine Experiment

The input to the system in this experiment is a sine wave over the range 0 to 4π. A sine wave was chosen because of the regularity of the input. The rate of change of a sine wave gradually changes over time. The pattern is repeated so there is the possibility the GRN could use this information to more easily solve the problem. For increased offsets the sine wave was wrapped around to ensure a continuous input with no

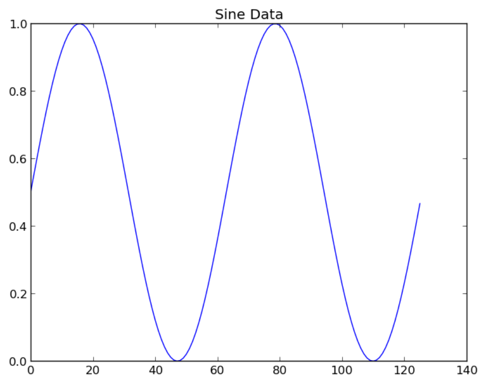

Fig. 5. The sine wave that is input into the system. An offset copy of the same sine wave is used to calculate the fitness.

large changes. The input is discretised using a step size of 0.1 to generate 126 separate inputs as shown in Figure 5. The samples are obtained using the following equation:

$$samples = \left\{ \frac{sin(n)+1}{2} \right\}_{n=0}^{4/pi}$$

The fitness value is calculated by summing the absolute difference between the P protein output and desired output over the 126 samples and is described by the equation:

$$fitness = \sum_{n=1}^{126} |p.out(n) - sample(n)|$$

B. Sine Results

The GRN outperformed linear regression for all of the sine experiments showing that it outperforms a basic deterministic method for fitting the data. It also outperformed copying the input except for offset 0 where it performed equivalently as copying the input is the best approach. Although larger synchronisation steps initially show a benefit when the offset is 0, this advantage quickly disappears for greater offsets. Sine offset 0, shown in Figure 6, highlights the need for a synchronisation step. Every step size matches the desired output except step size one, which plateaus after 50 generations.

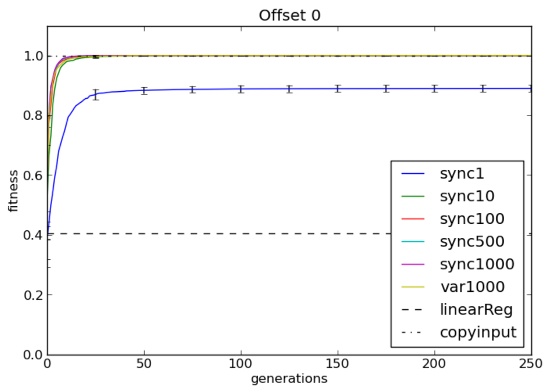

Fig. 6. Sine Offset 0 results.

The reason for such a good performance are examined in detail in Figure 7 which shows how the protein concentrations varied as the input was changed. The input and desired output are shown in black, the TF protein concentrations are shown in red, and the P protein concentrations are shown in blue. The best performing P protein is highlighted in green. It is clear that the TF protein has an inverse relationship with the input, as the input concentration increases, the TF protein decreases. The P proteins show different effects to the change in TF protein concentrations. The best result (in green) once again has an inverse relationship with the TF allowing it to directly match the input.

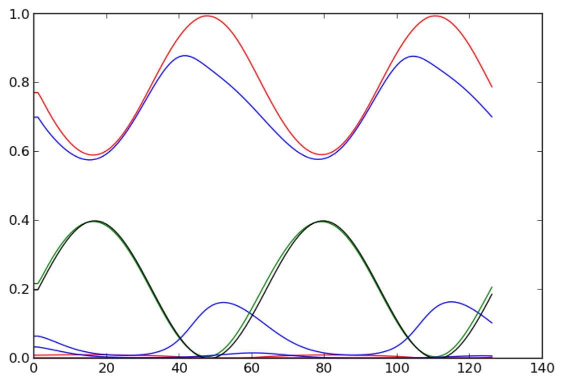

Fig. 7. Protein concentrations during the course of a run.

The results from offset 10 in Figure 8 generated an unexpected result as variable synchronisation step actually performed worse than any of the other fixed intervals. This result is interesting as no equivalent effect was seen in the random walk results, indicating that the regularity of the input had a negative effect on a variable sync step. The result indicates that the method used in the experiments for finding a stable state may be incorrect. If the system has reached a truly stable state then the results should be equivalent to completing 1000 iterations. The results for offset 20 in Figure 9 overlap substantially indicating that there is no statistically significant difference between the results.

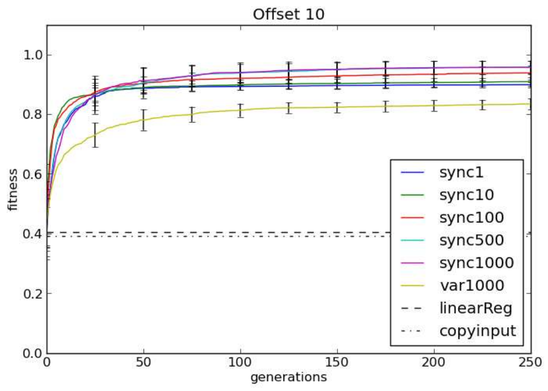

Fig. 8. Sine Offset 10 results.

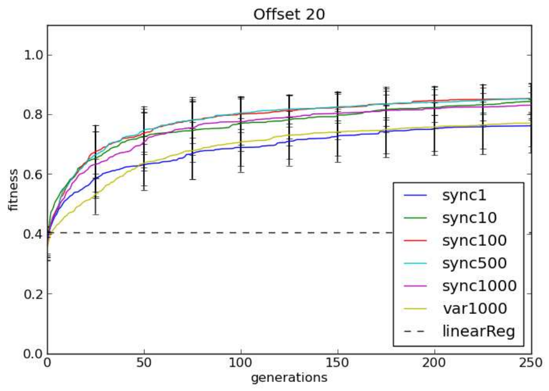

Fig. 9. Sine Offset 20 results, copy input not shown as it performed poorly.

C. Random Walk Experiment

In this experiment the input to the GRN consists of a succession of random steps. The step size for each input step was chosen from a uniform distribution between -0.02 and 0.02 for 1000 sample points. Such input provides no regularity or pattern so that the only way the GRN can match the signal is to use the previous input values. The aim of this experiment is to examine if the GRN is capable of using hysteresis (the dependence of a system not only on its current environment but also on its past environment) to solve the problem. The random walk input is shown in Figure 10. Again the experiments were carried out for 250 generations with a population size of 100 to examine if there was any improvement over a greater length of time. In order to baseline the results they were compared against a linear regression solution.

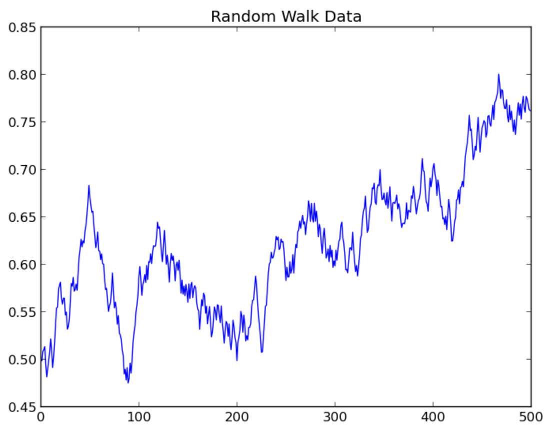

Fig. 10. The sequence of random steps that is input into the GRN.

D. Random Walk Results

The GRN outperformed linear regression and copying the input for all of the random walk experiments once again showing that it performed better than both baselines set for these experiments. The results for matching input and output shown in Figure 11 indicate that the synchronisation provides a clear benefit to performance as step size one plateaus after 50 generations and produces no additional gains after that.

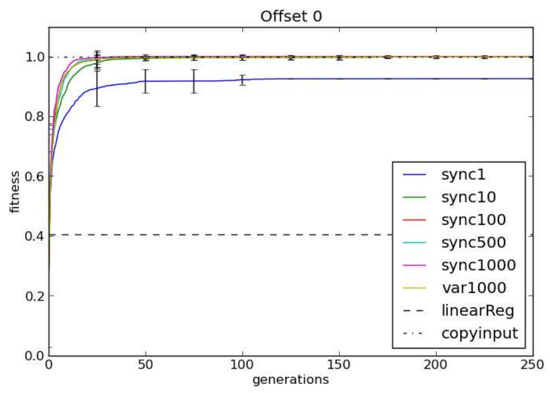

Fig. 11. Random walk offset 0.

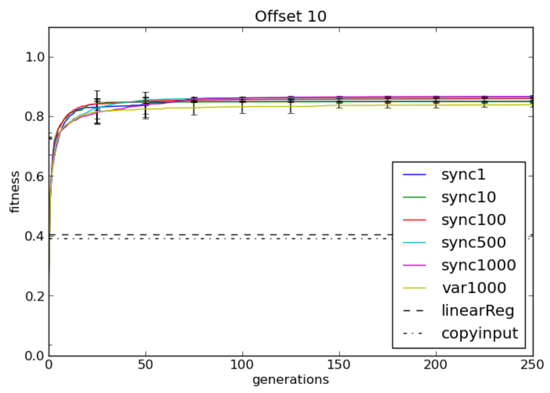

Fig. 12. Random walk offset 10.

The results for both the offset 10 and offset 20 in Figures 12 and 13 respectively contain considerable overlap of the results, indicating that any difference is not statistically significant.

VII. Conclusion

The pole balancing experiment examined in this work indicate that a many to one mapping between the physical system and the GRN model dramatically improves performance. One caveat of this finding was that the improvement increased logarithmically so that there were diminishing returns for increasing the step size. The offset experiments showed a similar improvement when the offset was zero but performed equivalently when the offset was increased. The results indicate that increasing the step size does not provide a benefit for all instances of a particular problem.

The experiments also showed that although a variable synchronisation step provided a mechanism for selecting the number of iterations, in some cases it performed worse than a fixed number of iterations. If the variable synchronisation step had correctly detected a stable state then the results should have been the same as using a fixed number of iterations. Such a result indicates that the mechanism for detecting a stable state requires improvement.

VIII. Future Work

An investigation into different methods for detecting a stable state, such as tracking the total change over several iterations

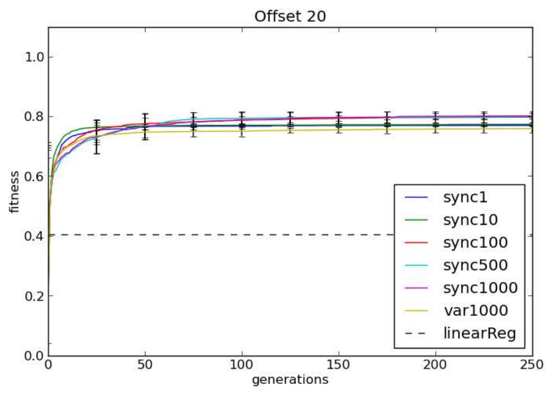

Fig. 13. Random walk offset 20, copy input not shown as it performed poorly.

or decreasing the change threshold, will be explored in future work.

IX. Acknowledgments

We would like to thank Science Foundation Ireland, the Financial Mathematics Computation Cluster and Andrea McMahon for her help during this project. We also wish to acknowledge the DJEI/DES/SFI/HEA Irish Centre for High-End Computing (ICHEC) for the provision of computational facilities and support. This work was funded by the SFI grants 08/RFP/CMS1115, 08/IN.1/I1868 and 08/SRC/FM1389.

References

[1] Wolfgang Banzhaf. On the dynamics of an artificial regulatory network. In *Advances in Artificial Life*, pages 217–227. Springer, 2003.

[2] Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *Systems, Man and Cybernetics, IEEE Transactions on*, (5):834–846, 1983.

[3] Jeff Hasty, David McMillen, Farren Isaacs, and James J Collins. Computational studies of gene regulatory networks: in numero molecular biology. *Nature Reviews Genetics*, 2(4):268–279, 2001.

[4] Miguel Nicolau, Marc Schoenauer, and Wolfgang Banzhaf. Evolving genes to balance a pole. In *EuroGP*, pages 196–207, 2010.

[5] Miguel Nicolau, Michael O'Neill, and Anthony Brabazon. Applying genetic regulatory networks to index trading. In *PPSN (2)*, pages 428–437, 2012.

[6] Ingo Rechenberg. *Evolutionsstrategie'94*, volume 1 of *Werkstatt Bionik und Evolutionstechnik*. Friedrich Frommann Verlag (Günther Holzboog KG), Stuttgart, 1994.

[7] Darrell Whitley, Stephen Dominic, Rajarshi Das, and Charles W Anderson. *Genetic reinforcement learning for neurocontrol problems*. Springer, 1994.