The Gene Regulatory Network: An Application to Optimal Coverage in Sensor Networks

Sanjoy Das
EECE Dept.
Kansas State University
Manhattan, KS
785 532-4642
sdas@ksu.edu

Praveen Koduru
EECE Dept.
Kansas State University
Manhattan, KS
praveen@ksu.edu

Xinye Cai
EECE Dept.
Kansas State University
Manhattan, KS
xinye@ksu.edu

Stephen Welch
Agronomy Dept.
Kansas State University
Manhattan, KS
785 532-7236
welchsm@ksu.edu

Venkatesh Sarangan
CS Dept.
Oklahoma St. University
Stillwater, OK
785 744-5672
saranga@cs. okstate.edu

ABSTRACT

This paper proposes a new approach for biologically inspired computing on the basis of Gene Regulatory Networks. These networks are models of genes and dynamic interactions that take place between them. The differential equation representations of such networks resemble neural networks as well as idiotypic networks in immune system. Although several potential applications have been outlined, an example, the problem of placing sensors optimally in a distributed environment is considered in detail. A comparison with NSGA-II suggest that the new method is able to accomplish near-optimal coverage of sensors in a network.

Categories and Subject Descriptors
I.2.8 [**Artificial Intelligence**]: Problem solving, control methods and search – *heuristic methods.*

General Terms
Algorithms, design, theory.

Keywords
Gene regulatory networks, sensors, heuristics.

1. GENE REGULATORY NETWORKS

Gene Regulatory Networks (GRNs) are models of genes and gene interactions at the expression level. As an emergent property of the network, simple interactions between the genes in the gene networks can give rise to surprisingly complex behavior of the network as a whole. Even small networks consisting of only one to four genes are able to function as Boolean logic gates, linear arithmetic units, circuit delay elements, differentiators, integrators, oscillators, coincidence detectors, and bi-stable devices [1, 2]. Consequently, one can see that GRNs are endowed with some processing capability.

GRNs have been traditionally modeled in a variety of ways. At the simplest level, Boolean networks have been used for this purpose. But more thorough models make use of differential equations. A popular nonlinear differential equation model is the S-formalism [*c.f.* 3]. The most general way to represent them is of the form,

$$\dot{x}_i = f(x_1, \ldots x_i, \ldots x_N, e_1, \ldots e_M) \tag{1}$$

where $f(\cdot)$ is a nonlinear function and x_i and $\dot{x}_i$ are the gene expression level and the rate of change of this level for the i^{th} gene in the network (i = 1, 2, ... N). The quantities e_1 through e_M are M environmental inputs.

The dynamic interactions within GRNs, ignoring environmental inputs, which suits our purpose well, may be formulated as in [1, 2],

$$\dot{x}_i = -\lambda_i x_i + \sigma\left(\alpha_i + \sum_j \beta_{ij} x_j + \sum_{i,j} \gamma_{ij} x_i x_j + \sum_{i,j,k} \delta_{ijk} x_i x_j x_k + \ldots\right) \tag{2}$$

Here the quantity $\lambda_i > 0$ is called the degradation rate. The quantities α, β, γ, and δ, indexed appropriately, are the additional network parameters. The function $\sigma(.)$ is a monotonically increasing nonlinearity bounded in [0, R], where R is a constant. Through these interactions, genes can either suppress or stimulate each other's expression levels, or even regulate them in more complex ways. Further biological details can be obtained from the literature on genomics [1, 2].

For example, in the molecular genetic model plant, *Arabidopsis thaliana*, three genes: TERMINAL FLOWERING1 (TFL1), APETALA1 (AP1), and LEAFY (LFY) play a special role in flowering. Two of them (AP1 and LFY) expressing themselves provokes plant commitment to flowering. The gene regulatory network formed from these three genes is shown in Figure. 1 (left) [4]. Figure. 1 (right) demonstrates a sample gene network topology that was algorithmically obtained [5] with three selected genes. It determines heading time in *Oryza sativa*, from two inputs, temperature (T) and photoperiod (P) [5].

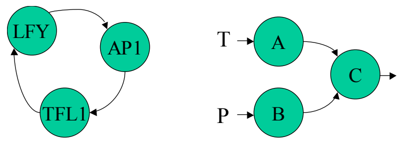

Figure 1. Example gene regulatory networks

2. GRN AS A DISTRIBUTED COMPUTATION

Although arguments within the nonlinearity in Equation 2 can be extended to indefinitely high order terms, the present paper restricts the highest to be quadratic, and ignores a few other terms, resulting in the simplified form,

$$\dot{x}_i = -\lambda_i x_i + \sigma\left(\beta_i x_i + \sum_{i,j} \gamma_{ij} x_i x_j\right) \tag{3}$$

Using the Equation 3, a GRN can be simulated numerically in a computer using Runge-Kutta methods [6]. The nonlinearity is modeled here as,

$$\sigma(z) = \max\left\{\frac{1-\exp(-z)}{1+\exp(-z)},\ 0\right\} \tag{4}$$

The above choice limits the range to [0, 1], which is suitable for most applications. The degradation rate λ_i serves as an inverse time constant. The values of the other network parameters can be appropriately assigned depending on application. Removing the higher order terms from Equation 2 restricts the possibility in which one gene can influence another to stimulation and suppression only. Noting that the gene expressions $x_i > 0$, when $\gamma_{ij} > 0$, the expression level of gene *j* promotes that of gene *i*. Conversely, when $\gamma_{ij} < 0$, the influence upon the latter is suppressive in nature instead.

The GRN is not necessarily fully connected; a nonzero γ_{ij} implying the existence of a pathway from the *j*th gene to the *i*th gene. Therefore, each gene receives inputs only from its immediate predecessors. Consequently, the communications between the genes in the GRN are limited, allowing the processing at each gene to proceed independently. In other words, the computations taking place in GRNs are distributed. This paper deviates from actual GRNs in one important aspect. Although feedback loops are present in actual GRNs, they do not occur between all possible genes. Ignoring this observation here extends the utility of the GRN paradigm for various applications – a useful tradeoff for biological fidelity.

2.1 Comparison with Other Networks

The differential Equation 2 has strong parallels to recurrent neural networks [7]. Both GRNs and neural networks involve nonlinearities; the key difference is the way these nonlinearities are applied. In recurrent neural networks, these nonlinearities (which are sigmoidal saturation ones), are used to restrict the outputs of the neurons to [0, 1], whereas the activations of the neurons are unbounded. The summations are applied following the nonlinearity. In contrast, in GRNs, the nonlinearity is applied to each gene following the summation terms. Because of this feature, the gene expression levels when the GRN stabilizes, is easy to foresee, by simply equating the left hand side of Equation. (3) to zero, yielding,

$$x_i = \frac{1}{\lambda_i}\sigma\left(\alpha_i + \beta_i x_i + \sum_{i,j} \gamma_{ij} x_i x_j\right) \tag{5}$$

This can be used effectively in an application, as in the sensor network problem considered here.

The GRN model also resembles idiotypic networks in immune systems [8, 13]. These networks portray antibody interactions, in terms of differential equations similar to Equation 3, where the left hand term is an antibody concentration. Again, there are key differences between idiotypic networks and GRNs. Models of idiotypic interactions do not include any higher order term, which makes GRNs more versatile than the latter for many applications. The interaction between the antibodies in idiotypic network is entirely suppressive, stimulation being based on external inputs only. Lastly, GRNs have a topological structure that these networks do not possess.

The topological differences between the three biological networks are summarized in Table 1.

Table 1. Comparison of the three biological networks

	Neural network	**Idiotypic network**	**GRN**
Nodal element	neuron activation	antibody concentration	gene expression
Edge element	synaptic weights	affinities	transcription
Interaction	excitation(+)/ inhibition(-)	suppression(-)	promotion(+)/ suppression(-)
Connectivity	sparsely/fully connected	fully connected	sparsely connected
Nonlinearities	present	present	present
Highest term	usually restricted	restricted	unrestricted

3. COVERAGE IN SENSOR NETWORKS

Our application is a simple version of the sensor network coverage problem [9, 10], which is introduced here. Given a set of *N* sensors, where each sensor s_i has a 2-dimensional coordinate $\mathbf{z}_i \in L$, where *L* is a 2-dimensional region on which sensors can be placed, called its location. For simplicity, we assume that this region is a unit square, i.e. $L \equiv [0, 1]^2$. Each sensor also has a radius, r_i. A sensor can be in two possible states, *ON* or *OFF*. Any point **p** inside *L* is said to be covered by a sensor s_i iff s_i = *ON* and the Euclidean distance between the sensor location and **p** is less than the sensor's radius. We write this as,

$$\mathbf{p} \in cov(i) \Leftrightarrow \| \mathbf{z}_i - \mathbf{p} \| < r_i \text{ and } s_i = ON \tag{6}$$

The number of active sensors is given by,

$$N_{active} = \left| \{ i \in (0, N] \mid s_i = ON \} \right| \tag{7}$$

The operator $|\cdot|$ denotes set cardinality. In a real world situation, it is desired to keep the number of active sensors as low as possible in order to minimize the battery consumption. At the same time, it is also desirable to maximize the total region in *L* that is covered by a sensor. We define the coverage as (see Figure 2),

$$C = \frac{\left| \{ \mathbf{p} \in L \mid \exists i \in (0, N] \,\&\, \mathbf{p} \in cov(i) \} \right|}{|L|}, \tag{8a}$$

or alternately as,

$$C = \frac{\left| \bigcup_{i=1}^{N} cov(i) \right|}{|L|} \tag{8b}$$

From a practical standpoint, the purpose of maximizing the coverage *C* is to allow the largest possible region inside *L* that can be reached by any sensor.

Thus, the goal of the sensor network coverage problem is to assign values *ON* or *OFF* to each s_i to attain maximum possible coverage while simultaneously keeping the number of active sensors, N_{active}, as low as possible to reduce power consumption. This problem can be formulated as a bi-objective optimization problem, with Equation 6 and Equation 7 providing the two objectives to optimize [9, 10]. Any well-known multi-objective optimization algorithm, such as NSGA-II [12] or FSGA [4] can directly be applied. Unfortunately, such an algorithm needs a centralized processor to run, rendering it infeasible in a practical situation. The GRN provides an attractive alternative to multi-objective optimization. This approach is described below.

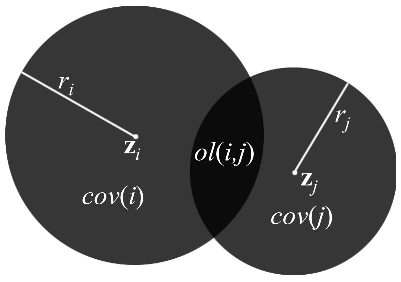

Figure 2. Two overlapping sensors

4. APPROACH

In the GRN approach, each gene in the regulatory network corresponds to a sensor. The gene expression levels x_i of the genes are decision variables. The network parameters are chosen such that sensors that should be turned *ON* eventually acquire high values of their corresponding gene expression levels. A threshold θ can be selected *a priori*, and any sensor with a higher value in its corresponding gene expression level can be turned *ON*. In other words,

$$x_i \begin{cases} \geq \theta \Rightarrow s_i = ON \\ < \theta \Rightarrow s_i = OFF \end{cases} \tag{9}$$

A lower θ causes a larger number of sensors to turn *ON*, which also has the effect of increasing coverage.

In this application, we have made an assumption that the power consumption of each active (*ON*) sensor is constant, independent of its radius. Although impractical in the real world, this assumption enables us to keep things simple enough to demonstrate the efficacy of the GRN approach. Other formulations of this problem can be dealt with by modifications of the GRN approach. Under the circumstances, it is desirable to bias the GRN such that sensors which cover larger regions are biased by the GRN algorithm towards higher expression levels. This is done by setting the values of the parameter β_i in Equation 3, corresponding to sensor *i* to be,

$$\beta_i = k_1 \times cov(i) \tag{10}$$

while setting β_j to zero for other sensors. If two active sensors are close together, there may be a common region that is simultaneously covered by both. For two sensors s_i and s_j (Figure 2), we define this overlapping region as,

$$ol(i, j) = cov(i) \cap cov(j) = \{ \mathbf{p} \mid \mathbf{p} \in cov(i) \,\&\, \mathbf{p} \in cov(j) \} \tag{11}$$

The covered region of each sensor as well as overlapping region between them can be computed directly from the locations $\mathbf{z}_i$ and $\mathbf{z}_j$ as well as their radii r_i and r_j as,

$$cov(i) = \pi \times r_i^2, \tag{12a}$$

$$ol(i, j) = r_i^2 \left(\cos^{-1} d_i \pm d_i \sqrt{1 - d_i^2} \right) + r_j^2 \left(\cos^{-1} d_j \pm d_j \sqrt{1 - d_j^2} \right) \tag{12b}$$

where d_i is obtained as shown (indexes can be changed for d_j),

$$d_i = \frac{r_i^2 + \| \mathbf{z}_i - \mathbf{z}_j \|^2 - r_j^2}{2 r_i \| \mathbf{z}_i - \mathbf{z}_j \|} \tag{12c}$$

Active sensors that are closely spaced together may have too much overlap. This introduces redundancy. In such a case, some of those sensors that do not provide additional coverage to the region that is not otherwise covered by another may be turned *OFF* to reduce their power consumption. The GRN biases these sensors towards lower gene expression levels through the parameter γ_{ij} in Equation 3. For any given sensor s_i, each neighboring sensor s_j, which has nonzero overlap with it, is allowed to exert a suppressive influence on it. Accordingly,

$$\gamma_{ij} = -k_2 \times ol(i, j) \tag{13}$$

Putting aside other considerations (such as sensor power consumption being proportional to the coverage radius), in order to elicit similar time characteristics in the gene expression levels, their degradation rates are set to a uniform constant value as shown,

$$\lambda_i = k_0 \tag{14}$$

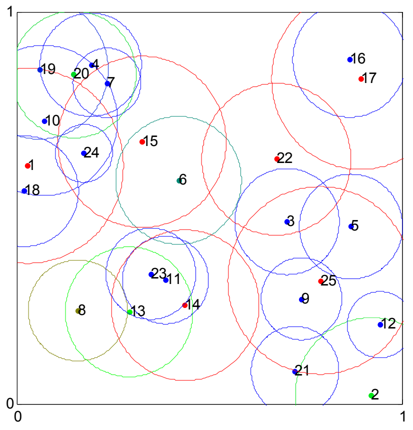

Figure 3. Placement of sensors for $N = 25$

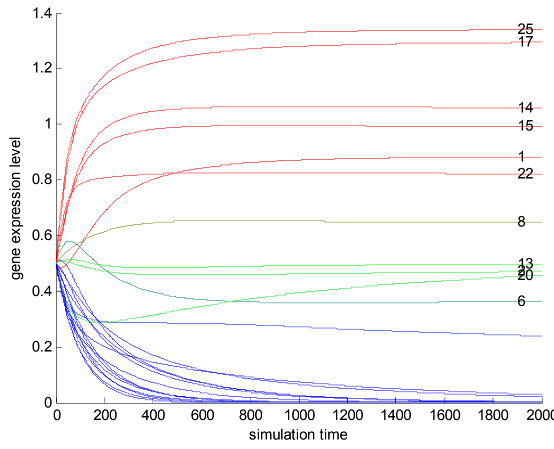

Figure 6. Time dependent gene expression for N=25

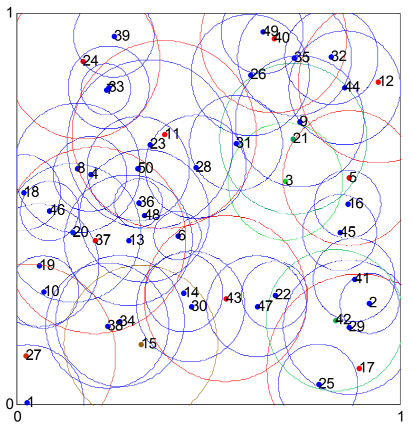

Figure 4. Placement of sensors for $N = 50$

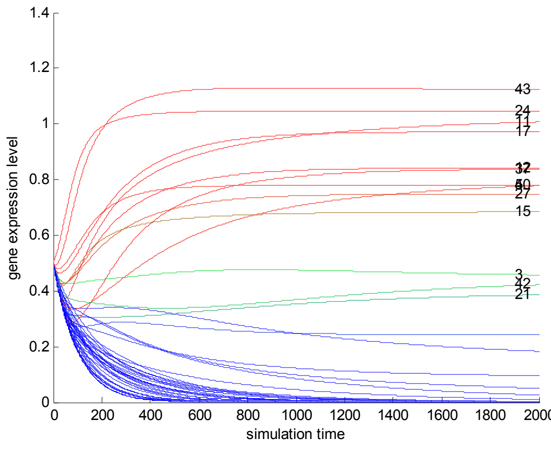

Figure 7. Time dependent gene expression for N=50

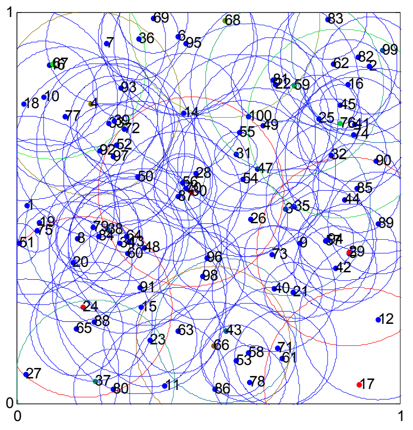

Figure 5. Placement of sensors for $N = 100$

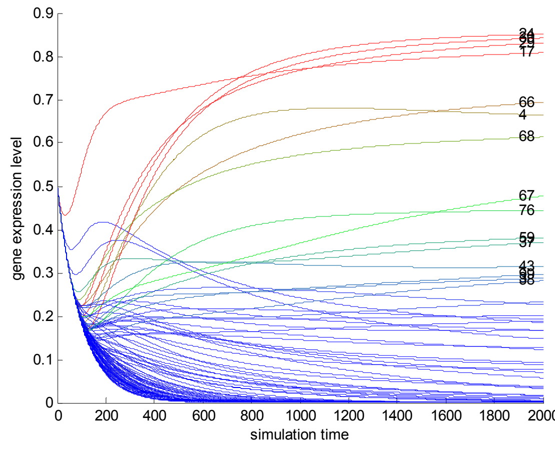

Figure 8. Time dependent gene expression for N=100

5. IMPLEMENTATION

The GRN constants associated with the sensor coverage problem are set to the following values, $k_0 = 0.01$, $k_1 = 1.8$, and $k_2 = 2.0$. The region L in the simulations is simply a unit square, $[0, 1]^2$. Although Equations 12a-c provides a direct method to compute the coverages and overlaps, a Monte Carlo simulation was used to estimate them since there are overlapping and covered regions that lie outside the unit square, L. It should be mentioned that early simulations producing nearly identical results suggest the benefit of Equations 12a-c, ignoring the above boundary conditions. The locations and radii of the sensors are generated randomly; the locations are uniformly distributed in the unit square while the radii follow a uniform distribution in the interval [0.05, 0.25]. All simulations are carried out for 2000 iterations with unit time increments although it is observed that the gene expressions tend to settle down at their final values at a much earlier stage (which, in a real world setting would translate to less computation, ergo diminished power consumption). All programs are implemented using Matlab on a Pentium-4 dual core processor.

5.1 Simulation Results

In order to test the effectiveness of the GRN approach for sensor networks of various sizes, the number of sensors was set to N = 25, 50, and 100. While the first problem serves as a demonstration of the capabilities of GRNs, the next two are reasonable for many sensor applications. Figures 3, 4, and 5 depict the locations and radii of the sensors for each value of N. The circles, centered around the locations, and with radii equal to that of the sensors, are color coded using the final expression levels of their corresponding genes. Sensors with higher expression levels are colored red, and those with least levels are colored blue. Intermediate ones are green. It has been seen that sensors with larger radii tended to acquire higher expression levels. Moreover, it is also observed that those sensors that are subsumed by other sensors in their vicinity eventually tend to acquire lower values, and tend to turn *OFF*.

Figures 6, 7 and 8 illustrate how the gene expressions, which are all initialized to the same expression level of 0.5, fluctuate with time. They are color coded in the same manner as before. For the smaller sensor networks (N=25, N=50), it is clear that the gene expressions asymptotically reach their stable values well ahead of time. Although this is the case with most sensors in the largest network (N=100), a few of them (such as sensors 66 and 67), do not stabilize readily. This highlights the decision making process involved within the GRN in deciding the appropriate values of the expressions.

This observation that complex decision making takes place is further reflected during the initial stages of the GRN approach, when several sensors whose gene expressions initially are observed to go down towards zero, which then move back up as their neighboring sensors are driven towards lower expression levels, leaving a larger gap to covered. Conversely, there are some sensors whose expressions, initially are driven upward, but eventually decrease when the GRN decides to impart higher expression levels to their neighbors. These observations indicate that the network coverage problem cannot be solved through simple heuristics.

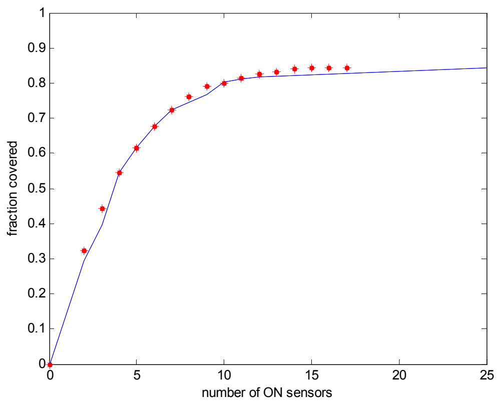

Figure 9. Comparison with NSGA-II for *N*=25 *(GRN – Blue line, NSGA-II – Red points)*

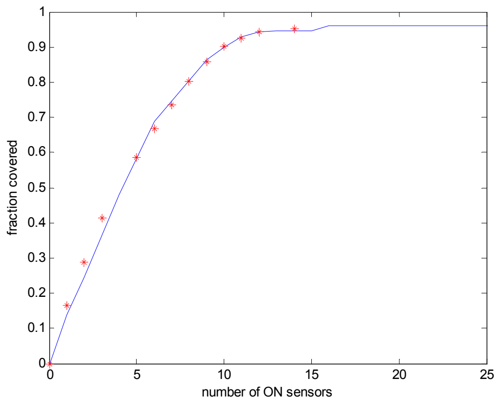

Figure 10. Comparison with NSGA-II for *N*=50 *(GRN – Blue line, NSGA-II – Red points)*

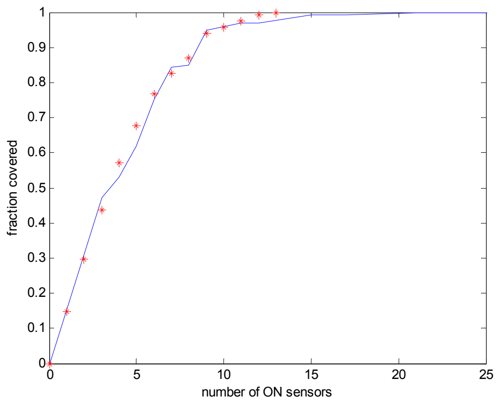

Figure 11. Comparison with NSGA-II for *N*=100 *(GRN – Blue line, NSGA-II – Red points)*

5.2 Comparison with NSGA-II

In order to evaluate the GRN approach, the sensor network problem is formulated explicitly as a bi-objective optimization problem and NSGA-II is applied. Although any other multi-objective methods would have suited this application, such as FSGA, which consistently outperforms NSGA-II [12] for standard benchmarks, the latter has been studied only for continuous optimization problems. Further, NSGA-II is a very popular multi-objective evolutionary algorithm.

The solutions in NSGA-II are encoded as binary strings of length *N*, which implements uniform crossover and mutation with rates of 50% and 3%. The population size in NSGA-II is 100. The choice of all these is standard for almost all applications, and also, as preliminary studies show, judicious for this application. The number of generations is equal to 100 if *N*=25 and *N*=50, and slightly higher at 120 if *N*=100.

The non-dominated front obtained by NSGA-II is compared with the outcome of the GRN approach. The equivalent of this front using the GRN method can be obtained by gradually increasing the threshold, θ, starting from zero, to get lower and lower active sensors N_{active}, i.e. reduced coverage *C*.

It is reasonable to expect that NSGA-II would produce a front that would be very close to the true Pareto front, if not equal to it. Therefore, the output of NSGA-II serves as a basis for gauging the performance of the proposed GRN scheme – the closer the result of the GRN approach to that of NSGA-II, the better is its performance. The results of this comparison are shown in Figures 9, 10, and 11 for each value of *N*. The solid blue lines correspond to the result produced by the GRN, with different thresholds. The red points show the Pareto front obtained by NSGA-II. It can be concluded from the comparisons that the two outputs of the GRN approach and NSGA-II are quite similar to one another. This illustrates that the GRN approach is able to perform as well as NSGA-II for this task. Noting that NSGA-II cannot be implemented in a distributed environment, hence the GRN approach could be a very good alternative to other methods of sensor network coverage.

5.3 Future Research in Sensor Coverage

Future research will apply this approach to other versions of sensor coverage. In particular, when it is necessary to ensure that each point is covered by at least two sensors, so that sensor failure will not leave any region uncovered. This fault-tolerant version of sensor coverage can be accomplished with GRNs by including the third order term in Equation 2, where the value of the network parameter δ_{ijk} is equal to the region that is overlapped by three sensors. This is a more modern approach for sensor network coverage, which also would show the utility of the more general GRN model in Equation 2. It should be noted that this term is absent in idiotypic and most neural networks. The introduction of this term also provokes the questions related to a network's stability, for GRNs can also exhibit limit cycles, as in the 24-hour circadian clock [1]. The authors also intend to explore the performance of GRNs with external targets. In such cases, the genes would be allowed to receive external inputs, and can be accomplished by using the constant term, α_i. Its analogy in real organisms is environmental factors, such as temperature and photoperiod that influences plant flowering time [4]. This approach can be extended for optimal dynamic coverage in sensors for moving target recognition by time varying values of α_i, which also finds a biological analogy as environmental inputs are constantly changing in nature.

6. DISCUSSION

This paper introduces the GRN as a computing paradigm and demonstrates its effectiveness for sensor coverage. Comparing it with NSGA-II has made it clear that the method works quite well, and unlike the latter, can be used to an adaptive self-configured sensor network.

The approach taken for the sensor network coverage problem is just an example GRN application. As the sensor coverage problem illustrates, GRNs can be used as a biologically inspired metaphor for discrete optimization. They can be applied for problems where the solutions can be represented as a sequence of ones and zeros, which can correspond to genes that are *ON* and *OFF* respectively. The parameters of the GRN can be determined through problem-specific heuristics in a manner similar to how they are done in Equation 10 and 13 for sensor network coverage. Although in this paper, each GRN produces in a single optimal solution, multiple solutions can also be generated from a single GRN by imparting small initial perturbations to the gene expressions. Such a scheme can pass well to an evolutionary algorithm. Instead of the conventional approach of evolving the solutions directly, the evolutionary operators can be applied to evolve the GRN parameters instead. Each GRN can in turn generate multiple solutions. This scheme would bear a resemblance to nature, where the organisms' genotype, rather than the phenotype is subject to Darwinian evolution.

GRNs can also be used as classifiers. A simple gradient descent algorithm can be applied to obtain the GRN parameters, which can also accept inputs. The classification decision would be based on assuming the combination of genes that express themselves in response to inputs. One advantage over conventional classifiers (such as neural networks, decision trees or support vector machines) is the fact that GRNs accept time-varying inputs. This can be used for more complex pattern classification. Further investigation is necessary in order to explore the possibility of using GRNs as such classifiers.

Lastly, GRNs can be used as decision support systems. GRNs offer the same advantage over knowledge-based systems as mentioned above, because they can also accept time-varying inputs. Further, GRN allow much flexibility in terms of topology, allowing feedforward as well as feedback connections, which are not possible with traditional IF-THEN-ELSE rules.

7. CONCLUSION

Biology has inspired several metaphors for computation. Darwinian evolution provided the background for evolutionary algorithms, while neuroscience gave rise to neural computation. More recently, distributed intelligence has yielded the Ant Colony Optimization (ACO) [14] and Particle Swarm Optimization (PSO) algorithms [15], while the vertebrate immune systems have produced a new class of algorithms in Artificial Immune Systems (AIS) [16]. Despite the large body of literature on genomics, GRNs have hitherto not been looked at for potential use in problem solving. Hence, further research is needed to explore the applicability of the GRN paradigm for real world problems.

8. REFERENCES

[1] S. M. Welch, J.L. Roe, S. Das, Z. Dong, R. He and M.B. Kirkham, *Merging genomic control networks with soil-plant-atmosphere-continuum (SPAC) models*. Agricultural Systems, 86: 243-274, 2005.

[2] S. M. Welch, J.L. Roe, and Z. Dong, *A genetic neural network model of flowering time control in Arabidopsis thaliana*, Agronomy Journal, 95: 71-81, 2003.

[3] L. F. A. Wessels, E. P. Van Someren, M. J. T. Reiders, A Comparison of Genetic Network Models, *Pacific Symposium on Biocomputing*, 508 – 519, 2001.

[4] P. Koduru, Z. Dong, S. Das, S. M. Welch, J. Roe, Multi-Objective Evolutionary-Simplex Hybrid Approach for the Optimization of Differential Equation Models of Gene Network, *IEEE Transaction on Evolutionary Computation*, in press.

[5] X. Cai, S. M. Welch, P. Koduru, S. Das, "Inferring Gene Regulatory Network Structures Using Genetic Programming and Particle Swarms, Proceedings, International Conference on Artificial Intelligence, Las Vegas, Nevada, 2007.

[6] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes in C++: The Art of Scientific Computing*. Cambridge University Press, UK, 1992.

[7] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1998.

[8] J. Pacheco, J. F. Costa, The Abstract Immune System Algorithm, *Proceedings of UC'07*, *6th International Conference on Unconventional Computation*, Eds: S.G. Akl, C.S. Calude, M.J. Dinneen, G. Rozenberg, H.T. Wareham, LNCS 4618, Springer-Verlag, Germany, 137 - 149, August 2007.

[9] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. Srivastava, Coverage Problems in Wireless Ad-Hoc Sensor Networks, *IEEE Infocom* 2001, Vol 3, pp 1380-1387, April 2001.

[10] Chi-Fu Huang and Yu-Chee Tseng, The Coverage Problem in a Wireless Sensor Networks, *Mobile Networks and Applications*, Volume 10, Number 4, August, 2005, pp. 519-528.

[11] S. Das, B. Panigrahi, Multi-Objective Evolutionary Algorithms, *Encyclopedia of Artificial Intelligence*, Eds: J. R. Rabual, J. Dorado, A. Pazos, Idea Group Publishing, in press.

[12] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, 6, 2, April, 2002.

[13] H. Altan and I.R. Cohen (Eds), *Theory of Immune Networks*, Berlin: Springer, 1989.

[14] M. Dorigo, V. Maniezzo, A. Colorni, Ant System: Optimization by a Colony of Cooperating Agents, *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 26, 1, 29-41, 1996.

[15] J. Kennedy and R. Eberhart, Particle Swarm Optimization, In *Proceedings of the IEEE International conference on Neural Networks,* 1942-1948, 1995.

[16] L. DeCastro and J. Timmis (Eds), *Artificial Immune Systems: A new Computational Intelligence Approach*, Springer, London, 2002.