Inference of Gene Expression Networks Using Memetic Gene Expression Programming

Armita Zarnegar, Peter Vamplew, Andrew Stranieri
School of Information Technology and Mathematical Sciences,
University of Ballarat,
P.O. Box 663, Ballarat, Victoria, Australia

{azarnegar@students.ballarat.edu.au,p.vamplew@ballarat.edu.au,
a.stranieri@ballarat.edu.au}

Abstract

In this paper we aim to infer a model of genetic networks from time series data of gene expression profiles by using a new gene expression programming algorithm. Gene expression networks are modelled by differential equations which represent temporal gene expression relations. Gene Expression Programming is a new extension of genetic programming. Here we combine a local search method with gene expression programming to form a memetic algorithm in order to find not only the system of differential equations but also fine tune its constant parameters. The effectiveness of the proposed method is justified by comparing its performance with that of conventional genetic programming applied to this problem in previous studies.

Keywords: Gene Expression Programming, Differential Equations, Gene Networks, Evolutionary Algorithm, Gene expression Profile, Microarray data .

1 Introduction

Microarray technology is a fast and versatile technique for exploring genome wide information such as gene function. A DNA microarray is a collection of microscopic DNA spots where each spot is a single gene attached to a solid surface (Tarca et al. 2006). DNA microarrays are commonly used for simultaneously monitoring the expression level of thousands of genes existing in a sample. They are used for a comparative genomic study such as cancer versus normal tissue (Dubitzky et al. 2003). Microarrays usually provide a static picture that shows the expression of many genes at a particular time in two different experimental samples. Recently researchers have started to use it for extracting a dynamic picture by getting different samples over time (Ideker et al. 2002; Wang et al. 2006). In this way, they are able to extract information about gene expression networks from the microarray data.

A gene expression (regulatory) network is a diagrammatic representation of gene expression over a period of time related to a situation, like the development of a disease. To obtain this network, usually multiple experiments must be carried out at different times or stages of a disease. Therefore, a dynamic picture can be extracted from microarrays which tells us about the developmental process of that condition through the gene regulatory network (which gene was first expressed and caused other genes to be expressed or inhibited in the second step and so on).

Finding gene regulatory networks is a complex task. The reason underlying this is the complicated nature of genetics. Variation in samples (or patients) makes a huge difference in the extracted network. Also, in reality, many genes interact with each other and this increases the complexity of the model exponentially. Moreover, current microarray technology produces noisy data. Additionally, in most cases, there are insufficient samples or records compared with the number of genes or variables, because of the expensive technology, which makes it even harder to build an accurate model. As a result of the above facts, finding gene regulatory networks is complex and nonlinear. This has become one of the major concerns in bioinformatics.

Many models have been proposed to represent gene expression networks. In Boolean networks (Akutsu et al. 1999), the gene expression level is either 0 or 1 and the difference in expression levels is not considered. Those methods which consider real value expression can be categorized into two groups; probabilistic methods such as Bayesian networks and deterministic methods such as temporal differential equations. Further information regarding different techniques for the reconstruction of gene regulatory network can be found in two recent surveys (Sehgal et.al 2008; Schlitt and Brazma 2007). Temporal differential equations are the most common technique used to build a gene expression network from time series data (Wang et al. 2006; Hallinan 2008).

Differential equations are a powerful and flexible model to describe complex relations among components. It is not easy to determine a suitable form of equations to represent the network, therefore, in some previous studies the form of the differential equations has been fixed (Sakamoto & Iba 2001). An S-system is a fixed form of differential equations that has been proposed as a model and the parameters are optimized by using a genetic algorithm.

In this paper, we deal with an arbitrary form of the right hand side of the system of differential equations to obtain a more flexible model, as shown in Equation 1:

$$\frac{dX_i}{dt} = f_i(X_1, X_2, ..., X_n) \quad (i = 1,2,..., n) \qquad \textbf{(1)}$$

where X_i is the expression level of the i-th gene (state variable) and n is the number of the genes (component) in the network. We use gene expression programming which is a new evolutionary computation technique to solve this problem.

2 Related Work

There have been several methods proposed for inferring gene expression or regulatory networks. Of particular interest to this paper are those approaches which use evolutionary computation methods to infer a model of differential equations from time series data.

Evolutionary computation is a particularly useful approach when a problem cannot readily be solved mathematically and we can not realistically look for an optimal solution but one or more good solutions are needed. Therefore, it is particularly suitable for the problem of inferring gene networks from microarray data. Different kinds of evolutionary computation techniques have been applied to this problem ranging from extensions of genetic algorithm to genetic programming, and differential evolution.

Sakamoto and Iba (2001) used genetic programming (Koza 1992) to solve this problem modeled by a system of differential equations. Solving the general form of a system of differential equations is very difficult so a fixed form, called the S-system (Savageau 1988), was used and the goal becomes simply to optimize the parameters in the fixed equations. An S-system is a type of power-law formalism. The concrete form of the S-system is as follows:

$$\frac{dX_i}{dt} = \alpha \prod_{j=1}^{n} X_j^{g_{ij}} - \beta \prod_{j=1}^{n} X_j^{h_{ij}} \quad (i = 1,2,..., n) \qquad \textbf{(2)}$$

where X_i is a state variable. The first term gives us all effect of increasing X_i whereas the second term gives the effect of decreasing X_i.

The first work which used genetic algorithms to solve the S-system was presented by Maki et.al. (Maki et al. 2001) There are other works which applied genetic algorithms to this problem such as a study by Morishita et al. (2003) which used an evolutionary algorithm to find parameters for an S-system representing a 5-node network. Kikuchi et al. (2003) at the same time reported a good result for the same number of nodes. Later on, in 2005, genetic programming was used to solve the S-system by Matsumura et.al. (2005) and appropriate solutions were obtained. Also, in 2005, for the first time differential evolution was used for this purpose by Noman and Iba (2005). Their work presented a high performance, however, in their study the number of genes was still limited to 5 and the model could not easily be scaled up for larger networks. The reason for this is the fact that the number of parameters in differential equations system is proportional to the square of the number of genes in the network. Therefore, when the number of genes increases the algorithms must simultaneously estimate a large number of parameters. This is why inference algorithms based on the differential equations model have only been applied to small-scale networks of less than five genes.

Evolutionary techniques were used along with other modeling approaches for gene regulatory networks. An example of that is a study by Eriksson and Olsson (2004) which used genetic programming to successfully solve a Boolean network of 20 genes.

In this paper, we try to solve the problem of inferring gene regulatory network modeled by a system of differential equations with an extension of the Gene Expression Programming (GEP) algorithm. GEP has been applied in many regression problems successfully. In particular, it were used previously in a similar application -solving elliptic differential equations- by Jiang et al. (2007).

Our algorithm exploits the effectiveness of GEP in finding the structure of gene regulatory network modeled by ordinary differential equations. It also uses a local search technique along with GEP for extra benefits. The combination of these methods, GEP as a global search for finding a function structure and a local search for fine tuning model parameters, results in a more powerful algorithm.

The combination of global search methods with problem-specific solvers is known as memetic algorithms (MAs) (Moscato and Norman 1998). The problem-specific solvers usually are implemented as local search heuristic techniques. The hybridization is meant to accelerate the discovery of an optimal solution or to reach a solution which is impossible to discover by either of the component methods (Krasnogor et al. 2006). So far, conventional genetic algorithms have mainly been used in MAs as the global search method, however, the scope of MAs is not limited to the genetic algorithms and in general any global search method can be used (Krasnogor, Smith 2005). Sakamoto and Iba (2001) used a local search algorithm along with genetic programming to obtain the constant parameters of the target function effectively. Here for the first time we have proposed a MA with GEP as the global search method. The Least Mean Square method (LMS) was used as the local search method. We have used the same data as were used in a previous study in the literature (Noman & Iba 2005) and compared the efficiency of our method with conventional genetic programming.

3 Gene Expression Programming

Gene Expression Programming (GEP) is a new form of genetic programming and was first introduced by Ferreira in 2001. Like genetic programming, it evolves computer programs but the genotype and the phenotype are different entities (both structurally and functionally) and because of this, performance is improved. It has been shown in experiments to converge faster than older genetic algorithms (Ferreira 2008). It also brings a greater transparency as the genetic operators work at the chromosome level (Wilson 2008).

GEP uses fixed length linear strings of chromosomes as the genotype, and the phenotype is in the form of expression trees which represents a computer program (Marghny & El-Semman 2005). These trees are then used to determine an organism's fitness. The decoding of GEP genes to expression trees implies a kind of code and a set of rules which are simple. The set of genetic operators applied to GEP chromosomes always produces valid expression trees (ET).

The most important application of GEP is in function finding and regression problems. Functions are the most important parts of a model. There are different approaches and methods for finding functions ranging from mathematical methods like logistic regression to artificial intelligence perspective via evolutionary computation. The latter method has the advantages of flexibility and generality as it is not limited to the assumption of linearity.

We use GEP to find the best form of differential equations from the observed time series of the gene expression. Although GEP is effective in finding a suitable structure, it is not so effective in optimizing the parameters of the formula such as constants or coefficients. This is the motivation for incorporating local search into GEP to build memetic gene expression programming. Local search methods can find the constant values and parameters effectively and GEP is known to be effective in finding function structures. This combination results in an effective algorithm which is highly capable in function estimation.

4 Memetic Gene Expression Programming for Gene Expression Networks

Here we present an algorithm designed to infer a gene expression network (gene regulatory network) from the observed time series data. As noted earlier, the problem can be modeled as a set of differential equations. We used a GEP algorithm to evolve the structure of the gene expression network and enhanced it by using the local search process to find the constant parameters of the equations more effectively.

The genes of gene expression programming are composed of a head and a tail. The head contains symbols that represent both functions and terminals, whereas the tail contains only terminals. For each problem, the length of the head *h* is chosen, whereas the length of the tail *t* is a function of *h* and *n* is the number of arguments in the function, and is evaluated by equation (3).

$$t= h\,(n-1)+1 \qquad (3)$$

Consider a gene for which the set of functions is F = {+, -, *, /, sqrt} and the set of terminals is T = {a,b }. In this case n = 2; if we choose an h = 6, then t = 6 (2 - 1) + 1 =7, thus the length of the gene is 6 + 7 = 13. One such gene is shown below:

.-.a./..sqrt.a.b.a.a.b.a.b

where "." is used to separate individual building elements, "sqrt" represents the square root function and a, b are variable names. The above is referred to as Kava notation, and the above string is called a K-expression (Li X et al. 2004).

5.1 Fitness Function

In general, the genetic network inference problem is formulated as a function optimization problem to minimize the following sum of the squared relative error and the penalty for the degree of the equations:

$$f=\sum_{i=1}^{n}\sum_{k=0}^{T-1}\left(X_i'(t_0+k\Delta t)-X_i(t_0+k\Delta t)\right)^2+\sum_{j=0}a_jb_j \qquad (4)$$

$$\left(\begin{array}{ll} t_0 & : \text{the starting time} \\ \Delta t & : \text{the step size} \\ n & : \text{the number of the components in the network} \\ T & : \text{the number of the data points} \end{array}\right)$$

where $x_i(t_0+k\Delta t)$ is given target time series (k=0,1,..., T-1). $x'(t_0+k\Delta t)$ is the time series acquired by calculating the system of differential equations represented by a GEP chromosome. All of these time series are calculated by using the Runge-Kutta method. This fitness function has often been used in previous studies in GP, for example by Samakato and Iba (2001). The problem of inferring gene networks based on the differential equations has several local optima because the degree of freedom of the model is high. Therefore, a penalty function has been introduced by Kimura et al. (2004). This penalty function, which is the second part of the fitness function, encourages low degree solutions. a_j is the penalty coefficient for the *j*th degree and b_j is the sum of the absolute values of coefficients of *j*th degree.

5.2 Local Search for the Local Optimizations of the Model

GEP is capable of finding a desirable structure effectively, but it is not very efficient in the optimization of the constant parameters as it works on the basis of the combination of randomly generated constants. Thus, we use the least mean square (LMS) method to explore the search space in a more efficient way. To be more specific, some individuals are created by the LMS at some intervals of generations. Thus we use the LMS method to drive the coefficient of the expression of the right-hand sides of the system of differential equations.

Consider the expression approximation in the following form:

$$y(x_1,\ldots,x_L)=\sum_{k=1}^{M}a_kF_k(x_1(i),\ldots,x_l(i)) \qquad (5)$$

where $F_k(x_1(i),\ldots,x_l(i))$ is the basis function, $x_1,\ldots,x_L$ are the independent variables, y($x_1,\ldots,x_L$) is the dependent variable, and M is the number of the basis functions.

Let *a* be the coefficient vector, and χ^2 as follows:

$$\chi^2 = \sum_{i=1}^{N} (y(i) - \sum a_k F_k(x_1(i),..., x_l(i)))^2 \qquad \textbf{(6)}$$

The purpose of the local search is to minimize the function in Equation 6 to acquire *a*. *N* is the number of data points. Let *b* be the vector *y(1),...y(N)* and *A* be a *N*N* matrix described as follows:

$$\begin{bmatrix} F_1(x_1(1),...,x_L(1))...F_M(x_1(1),...,x_L(1)) \\ F_1(x_1(2),...,x_L(2))...F_M(x_1(2),...,x_L(2)) \\ ... \\ F_1(x_1(N),...x_L(N)...F_M(x_1(N),...,x_L(N)) \end{bmatrix} \qquad \textbf{(7)}$$

y(i) for the *i*-th equation of the system is calculated as follows:

$$y(i) = \dot{X}_j\big|_{t=ti} = \frac{x_j(t_i + \Delta t) - x_j(t_i)}{\Delta t} \qquad \textbf{(8)}$$

Then the following equation should be satisfied to minimize equation 6.

$$(A^T A).a = A^T b \qquad \textbf{(9)}$$

a can be acquired by solving Equation 9.

5.3 Overall Algorithms

- The GEP evolution begins with the random generation of linear fixed-length chromosomes for individuals of the initial population.
- In the second step, the chromosomes are translated into expression trees and subsequently into mathematical expressions, and the fitness of each individual is evaluated based on the formula presented in Equation 4 by using the Runge-Kutta method.
- Local search is applied on individuals at some interval generations
- The worst individuals are replaced in the population with the improved individuals generated above.
- Selection is done with tournament selection and then genetic recombination

The above steps repeat until there is no further improvement in the fitness function.

The local search algorithm has been applied in two different ways. In the first way, it has been used only for the best individuals in each generation, and in the second approach it has been used on the whole generation at some intervals. The result of the second method was better than the first method; therefore, the reported results are based on the second method of applying the local search procedure.

5 Experiments

To confirm the effectiveness of the proposed algorithm, we have used a small network model with four sets of time series data with different initial values. The number of network components is considered to be five.

Among those four experiments, here we present results for one which is the most complicated example. Fig.1. shows the gene network used in this experiment.

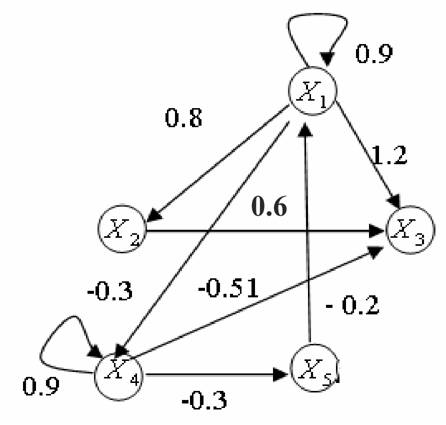


Fig. 1. A sample of weighted Gene Regulatory Network

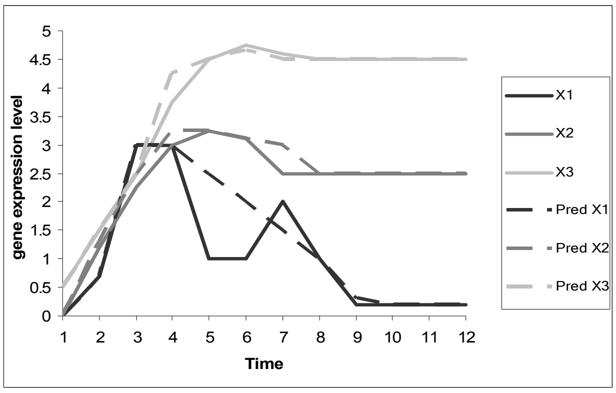
A weighted network was proposed to represent gene networks (Weaver et al. 1999). Each node is a gene and an arrow indicates a regulatory relation between two elements (gene). Negative values show a suppression relation and positive values show promotion.

To account for the stochastic behavior of GEP, each experiment was repeated for 20 independent runs, and the results were averaged. Table 1 lists the parameter values used for these runs.

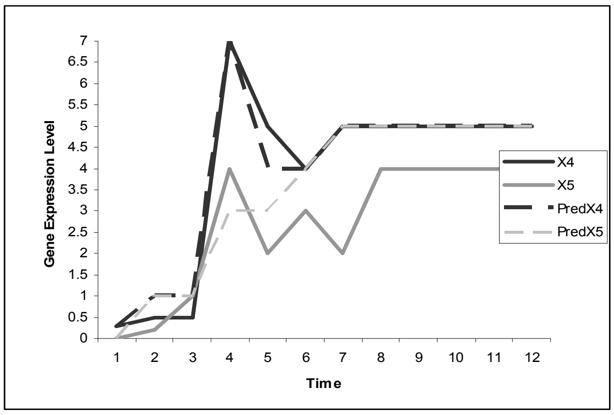
Table 1. General settings of our algorithm

Number of generation	500
Population size	100
Mutation rate	0.044
One-point recombination rate	0.2
Tow-points recombination rate	0.2
Gene recombination rate	0.1
IS transition rate	0.1
RIS transition rate	0.1
Gene transposition rate	0.1
Function set	+ - * /
Terminal set	α

Fig 2a and Fig 2b show the observed expression levels of the five components (gene) of the network and the predicted level produced by our method.



(a)



(b)

Fig. 2. Predicted versus actual gene expression levels for the best model obtained

The effect of local search on the performance of the algorithm is presented in Fig. 3. The local search was applied in two different ways; in the first one it was applied to the best individual of the generation and in the second it was applied to the whole population. The first approach rarely improved the performance, but the second approach significantly improved the fitness of average individuals in the population, especially in the early stages of evolution.

The reported result is based on the second approach of applying local search. It can be seen that on average the memetic system using both GEP and LMS achieves superior fitness levels compared to the system using GEP alone.

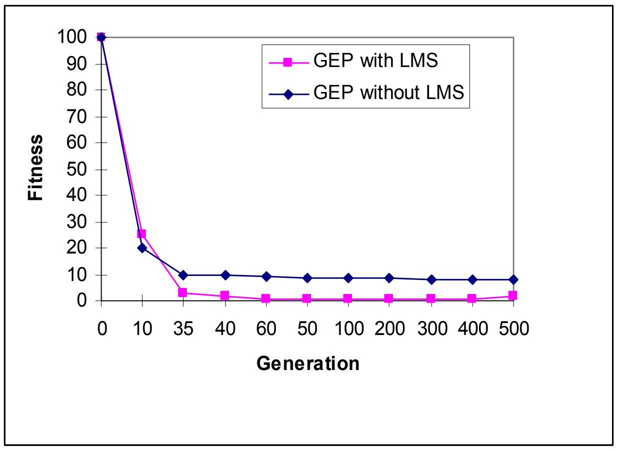


Fig. 3. Effect of the local search

We have also compared our algorithm with the conventional GP algorithm. For this purpose we have used GPLAB (MATLAB toolbox for genetic programming) with default parameter values. The result is presented in Figure 4. It shows that the proposed method has a faster convergence rate by an index of 100 compared to the conventional GP.

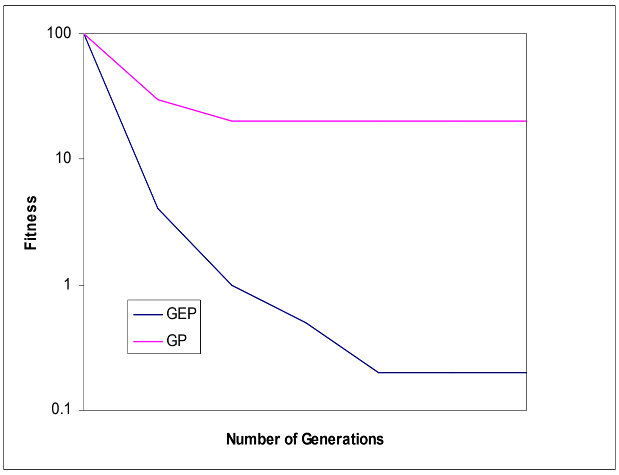


Fig.4. Performance comparison of the proposed GEP against GP

5.1 Effect of Noisy Data

We introduced artificial noise to the data to find out the robustness of our method. Usually in microarray data the most common noise is missing values. Therefore, we considered this type of noise here. We started with one missing variable per sample (2 percent noise) and then increased the amount of missing variables to 10 percent. The effect of such noise is presented in Table 2.

In the second experiment, we tested the effect of Gaussian noise on the data by perturbing a certain value x_i with a random number drawn from a Gaussian distribution $N(0,\sigma_i)$ by $x_i' = x_i + \sigma_i N(0,1)$. We present the correlation coefficient (*r*) that quantifies similarity between predicted values and observed ones as the measure of robustness of the algorithm in the presence of noise. Table 3 shows the result of applying noise to the gene expression values.

Output	***r***
Output without noise	0.891
Output with 2% noise	0.846
Output with 10% noise	0.798
Output with 20% noise	0.702

Table. 2. Effect of noise with adding missing values

Output	***r***
Output without noise	0.891
Output with 2% noise	0.888
Output with 10% noise	0.863
Output with 20% noise	0.801

Table. 3. Effect of Gaussian noise

The results in Table 2 and Table 3 show that the noise in the form of missing values affects the algorithm more than Gaussian noise.

The proposed system presents a robust behavior in the presence of noise, along with good performance. To compare the robustness of this algorithm in the presence of noise and also further investigation of the type of noise on our GEP system, we investigated Gene Expression Programming (GEP) literature. It has been said that GEP is a robust method in the presence of noise, although, there is not enough literature available on the effect of different types of noise on GEP systems. The only evidence of this type of work is a study by Lopez and Weinert (2004). In this work they used a simple form of random noise on each value and obtained a good result. Therefore, we decided to review the effect of noise on genetic programming (GP) algorithms as GEP can be considered to be an extension of GP.

Typically, the fitness function for the regression problems is based on a sum-of-errors, involving the values of the dependent variable directly calculated from the candidate expression. Although this approach is extremely successful in many circumstances, its performance can decline considerably in the presence of noise. Therefore, in a study by Imada and Ross (2008) it was suggested to use feature-based fitness function in which the fitness scores are determined by comparing the statistical features of the sequence of values rather than actual values themselves. This sort of fitness function can be considered for future research in improving the algorithm in the presence of noise.

6 Conclusion and Future Work

Recently, evolutionary computation methods have been used for model-based inference of gene regulatory networks. This is now a very challenging task in the bioinformatics area. In this work, we have investigated the suitability of Gene Expression Programming (GEP) for this problem. We have also proposed a memetic version of GEP which uses LMS as the local search procedure to improve the quality of solutions. The experimental results reported in this paper, using synthetic gene expression data, show that the proposed memetic GEP algorithm has a strong capability to find a suitable combination of constants and function structures. The constant creation method (local search) applied to the best individual of the generation can seldom improve them, however, when it is applied to the whole population it can significantly improve the fitness of average individuals in the population, especially in the early stages of evolution.

The proposed GEP can be further examined with other local search methods to more effectively fine tune parameters. It is also vital to increase the number of genes in the network to scale up this method as much as possible. In reality the gene regulatory network usually has more than 10 components. To the best of the authors' knowledge, existing evolutionary techniques can not deal with this number of components considering real gene expression values. Partitioning is a possible solution to scale up these methods. There are some partitioning methods which have been previously used with other evolutionary algorithms (Kimura et al. 2004) and have improved their scalability dramatically.

Also, in order to study the effect of real noise on our algorithm, the noise in the real data needs to be mathematically modelled. Then, it is possible to investigate the effect of real noise on our algorithm. The only part of the noise in our study which has a corresponding part in nature is the missing values. Modelling of noise in the form of mutated values is subject to further investigation of the distribution of noise in real microarray data.

References

Akutsu, T., Miyano, S. and Kuhara, S. (1999): Identification of Genetic Networks from a Small Number of Gene Expression Patterns under the Boolean Network Model. In *Proceedings of Proceeding of Pacific Symposium on BioComputing*, pp. 17-28.

Dubitzky, W., Granzow, M., Downes, C.S. and Berrar, D. (2003): *A Practical Approach to Microarray Data Analysis*, Springer.

Erikson, R. and Olsson, B. (2004): Adapting genetic regulatory models by genetic programming. *bioSystems 76: 217-227*

Ferreira, C. (2001): Gene Expression Programming: A New Adaptive Algorithm for Solving Problems. *Complex Systems* 13(2): 87-129.

Ferreira, C. (2008): What is Gene Expression Programming? from*: Http://www.gene-expression-programming.com* Accessed Jun 2008.

Hallinan, J. (2008): *Chapter4: Gene Networks and Evolutionary Computation. Computational Intelligence in BIOINFORMATICS*. Fogel, G. B., Corne, D. W. and Pan, Y., IEEE Press.

Ideker, T., Ozier, O., Schwikowski, B. and Siegel, A.F. (2002): Discovering regulatory and signaling circuits in molecular interaction networks. *Bioinformatics* **18**: Suppl 1: S233–S240.

Imada, J.H. and Ross, B.J. (2008): Using Feature-based Fitness Evaluation in Symbolic Regression with Added Noise. In *Proceedings of Genetic And Evolutionary Computation Conference (GECCO)*, Atlanta, GA, USA, pp. 2153-2158, ACM.

Jiang, D., Wu, Z. and Kang, L. (2007): Parameter Identification in Differential equations by Gene Expression Programming. In *Proceedings of International Conference on Natural Computation (ICNC)*, IEEE.

Kikuchi, S., Tominaga, D., Arita, M., Takahashi, K. and Tomita, M. (2003): Dynamic modeling of genetic networks using genetic algorithm and S-system. *Bioinformatics* 19(5).

Kimura, S., Ide, K., Kashihara, A., Kano, M., Hatakeyama, M., Masui, R., Nakagawa, N., Yokoyama, S., Kuramitsu, S. and Konagaya, A. (2004): Inference of S-system models of genetic networks using a cooperative coevolutionary algorithm. *Bioinformatics* 21(7): 1154-1163.

Koza, J. (1992): *genetic programming on the programming of computers by means of natural selection*. Cambridge, MA, MIT Press.

Krasnogor, N. and Smith, J. (2005) A Tutorial for Competent Memetic Algorithms: Model, Taxonomy and Design Issues. *IEEE Transactions on Evolutionary Computation* A, NO. B, CCC 200D

Krasnogor, N., Aragón, A. and Pacheco, J. (2006): *MEMETIC ALGORITHMS*. Metaheuristic Procedures for Training Neutral Networks, Springer US.

Maki, Y., Tominaga, D., Okamoto, M., Watanabe, S. and Eguchi, Y. (2001): Development of a system for the inference of large scale genetic networks. *Pacific Symposium on Biocomputing* **6**: 446-458.

Marghny, M.H. and El-Semman, I.E. (2005): Extracting Logical Classi□cation Rules With Gene Expression Programming: Microarray Case Study. In *Proceedings of AIML 05 Conference*, Cairo, Egypt.

Matsumura, K., Oida, H. and Kimura, S. (2005): Inference of S-system models of genetic networks by function optimization using genetic programming. *Transactions of the Information Processing Society of Japan* 46(11): 2814-30.

Moscato, P. and Norman, M. (1989). A competitive-Cooperative Approach to Complex Combinatorial Search *(No. C3P-790): Caltech Concurrent Computation Program.*

Noman, N. and Iba, H. (2005): Inference of gene regulatory networks using s-system and differential evolution. In *Proceedings of Proceedings of the 2005 conference on Genetic and evolutionary computation*.

Sakamoto, E. and Iba, H. (2001): Inferring a System of Differential Equations for a Gene Regulatory Network by using Genetic Programming. In *Proceedings of Evolutionary Computation,* 2001, pp. 720 - 726.

Savageau, M.A. (1988): Introduction to S-systems and the underlying power-law formalism. *Math. Comput. Modell* **11**: 546–551.

Sehgal, M. S., Gondal, I., Dooley L. (2008): Computational modelling strategies for gene regulatory network reconstruction.*Computational Intelligence in Medical Informatics :* 207-220, Springer

Tarca, A.L., Romero, R. and Draghici, S. (2006): Analysis of microarray experiments of gene expression profiling. *American Journal of Obstetrics and Gynecology* **195**: 373–88.

Wang, Y., Joshi, T., Zhang, X.-S., Xu, D. and Chen, L. (2006): Inferring gene regulatory networks from multiple microarray datasets. *Bioinformatics* **22**(19): 2413-2420.

Weaver, D.C., C.T.Workman and G.D.Storm (1999): Modeling Regulatory Networks with Weight Matrices. In *Proceedings of Pacific Symposium on Bioinfomratics*, pp. 112-123.

Wilson, S.W. (2008): Classifier Conditions Using Gene Expression Programming (No. IlliGAL Report No. 2008001): University of Illinois at Urbana-Champaign, USA.