An Adaptive GA–PSO Approach with Gene Clustering to Infer S-system Models of Gene Regulatory Networks

WEI-PO LEE* AND YU-TING HSIAO

Department of Information Management, National Sun Yat-sen University, Kaohsiung, Taiwan
*Corresponding author: wplee@mail.nsysu.edu.tw

The construction of gene regulatory networks from expression data is one of the most important issues in systems biology research. However, building such networks is a tedious task, especially when both the number of genes and the complexity of gene regulation increase. In this work, we adopt the S-system model to represent the gene network and establish a methodology to infer the model. Our work mainly includes an adaptive genetic algorithm-particle swarm optimization hybrid method to infer appropriate network parameters, and a gene clustering method to decompose a large network into several smaller networks for dimension reduction. To validate the proposed methods, different series of experiments have been conducted and the results show that the proposed methods can be used to infer S-system models of gene networks efficiently and successfully.

Keywords: systems biology; gene regulatory network; S-system; genetic algorithm; particle swarm optimization; gene clustering

Received 7 April 2010; revised 7 January 2011
Handling editor: Ying Tan

1. INTRODUCTION

Constructing gene regulatory networks (GRNs) has been identified as one of the most important issues in systems biology research [1]. With the current experimental tools and techniques, researchers are now able to simultaneously measure and to record the relevant information of multiple genetic reactors and their interactions. They can gradually model the overall systems from the experimentally measured time-series data, and then infer some unknown interactions between the reactors accordingly. Traditionally, to reconstruct GRNs manually takes a considerable amount of time, therefore an automated procedure (i.e. reverse engineering) is advocated [1, 2]. This procedure involves altering the gene network in some ways, observing the outcome and using mathematics and logic (i.e. computational methods) to infer the underlying principles of the network. The present work is to establish a practical methodology for such a procedure of network inference.

In the above-mentioned network reconstruction procedure, the most important steps are selecting the network model and fitting the network’s structural parameters to the available data. Depending on the biological level to be studied, many gene regulation models have been proposed, and they range from very abstract (involving Boolean values only) to very concrete (including fully biochemical interactions with stochastic kinetics). The abstract models are mathematically tractable, which provide the possibility of examining large systems, but they are inherently static and cannot infer networks with feedback loops. In contrast, the concrete models are more suitable in simulating the biochemical reality, but due to their computational complexities, these models can only be applied to small systems. With a chosen network model, several optimization techniques have been proposed to determine network parameters and structures from the available data [3, 4].

Among the many computational methods, evolutionary algorithms (EAs), especially genetic algorithms (GAs), have been widely used to solve many optimization problems with good results, and can be used for the network modeling task here. EAs are population-based approaches that evaluate many solutions simultaneously in the search space. They are likely to find a global solution for a given problem. However, these algorithms involve stochastic search procedures and require many iterations to converge to a good solution. Therefore, the speed of convergence becomes an important factor to be
considered. Recently, a new population-based optimization technique, namely particle swarm optimization (PSO, [5]), was proposed as an alternative to traditional EAs. PSO has some attractive characteristics. In particular, it has memory, so that knowledge of good solution can be retained by all particles (solutions). Some performance comparisons between traditional EAs and PSO have been made (for example, [6, 7]), underlining the reliability and the convergence speed of both methods. However, the result tends to be inconclusive: each of the algorithms has shown better performance than the other for some particular applications. Consequently, hybrid techniques were proposed to effectively exploit the qualities and uniqueness of the two methods. It is now generally agreed that the hybrid methods can lead to further performance improvement [8–10].

In this work, we adopt the non-linear S-system model to represent GRNs and develop an adaptive GA–PSO hybrid method that exploits the advantages of both GA and PSO to infer networks from expression gene profiles. We also propose to take the forms of numerical constraints to transfer and integrate domain knowledge into the search-based method to derive meaningful solutions. In addition, for large networks that are difficult to infer from the expression data by a search-based approach directly, we devise a divide-and-conquer strategy based on the principle of dimension reduction to deal with the scalability problem. In this strategy, a gene clustering method with some data analysis techniques for feature extraction is used to decompose GRNs and then the target networks can be inferred hierarchically. To validate the proposed methodology, different sets of experiments have been conducted. The results show that our methods can infer the S-system models of GRNs efficiently and successfully.

2. BACKGROUND AND RELATED WORK

As mentioned above, gene regulatory models can be categorized into two major types: those that use discrete and continuous variables, respectively [3, 4]. The first type of GRN model assumes that genes only exist in discrete states. This approximation is usually implemented by Boolean variables in which the gene is in either on or off state. Models of this type include Boolean networks and Bayesian networks. They are easy to simulate and thus less computationally taxing, but admittedly, they are not able to capture certain system behaviors.

The second type of model uses continuous variables to simulate fully biochemical interactions with stochastic kinetics [3, 4]. One of the more popular continuous variable models is the neural network-based model, and among which the recurrent neural network (RNN) model is the most successful one. This model is continuous in time, and uses a transformation function to transfer the inputs into shapes similar to those observed in natural processes. Although continuous variable models can more accurately model the system dynamics of gene networks, this modeling approach can only be applied to relatively small systems due to their computational complexity.

For the above models, many computational methods have been proposed to infer the network parameters correspondingly. For example, the GA has been used to evolve network parameters in [11], and the back-propagation learning method has been used to infer parameters for RNN-based gene networks in [12]. Among these methods, the most relevant works are [11, 13]. Similar to the two works, we also employ a population-based method to search the most appropriate parameters for the gene model. However, our work differs from them in several ways. Considering the optimization techniques, we develop a hybrid approach of GA and PSO to derive network parameters, but the authors of [11, 13] used a differential evolution technique and an orthogonal GA, respectively, to infer network parameters. In addition, to control the search direction in the solution space, we propose to take knowledge constraints to guide the search, while in [11, 13] the authors defined a penalty term to avoid invalid exploration. Most importantly, for large networks, we develop a gene clustering method to deal with the scalability problem but it has been ignored in both of the above works.

Our approach involves both GA and PSO techniques. In general, a solution in GA is represented as a fixed-length string of genes; each of them is regarded as carrying a genetic feature. A fitness function is defined to evaluate the solutions to determine the fitters. GA is operated as a cyclically iterative mechanism, which includes a sequence of selecting parents and creating children, after the initialization phase. The selection phase involves probabilistically choosing individuals from the current population as parents to generate offspring to form a new population, and the recreation phase is to apply genetic operators to the selected individuals for the generation of new ones. PSO is also a population-based technique. It tries to mimic the goal-seeking behavior of biological swarms. The basic PSO algorithm contains a set of particles and operates in an iterative manner. Each particle is characterized by its position and velocity, and it moves in the solution space. The position of each particle represents a potential solution and is evaluated by a predefined fitness function. During the iterative search process, each particle remembers its previous best position and the best position of any particle in the swarm. Then the particle uses the position information to modify its position and velocity, and continues its movement in the search space. A comprehensive survey of the PSO algorithms and their applications can be found in [5, 14].
3. INFERRING S-SYSTEM MODEL OF GRN

3.1. Network model and performance measurement

GRNs are complex biological systems. The network structure is an abstraction of the chemical dynamics of this system, describing the manifold ways in which one substance affects all the others to which it is connected. The network nodes can be regarded as functions obtained by combining basic functions upon the inputs. As mentioned previously, differential equations, including linear ordinary differential equations (ODEs) and non-linear power law differential equations, have often been used to describe the regulatory relationships between genes. In general, the change in the expression level of a gene at a certain time is characterized by a function that takes the regulatory influence (activation or inhibition) of other genes into account. It can be described as

\[
\frac{dx_i}{dt} = f_i(x_1, x_2, \ldots, x_N, p, u).
\]

In the above equation, \(x_i\) (1 \(\leq\) \(i\) \(\leq\) \(n\)) is the expression level of gene \(i\) at time \(t\), \(N\) is the number of genes, \(p\) is the parameter set of the system and \(u\) is an external perturbation to the system. The function \(f_i\) can be linear, piecewise linear, pseudo-linear or continuously non-linear, each describing the system dynamics with a different level of complexity.

The most popular and well-researched ODE model, the S-system, is a power law model. It consists of a particular set of tightly coupled ODEs, in which the component processes are characterized by power law functions. This model has been extensively studied, and is considered suitable to characterize the gene regulations [3, 4]. Therefore, in this work we adopt this model to represent GRNs. An S-system model has the following form:

\[
\frac{dx_i}{dt} = \alpha_i \prod_{j=1}^{N} x_j^{g_{i,j}} - \beta_i \prod_{j=1}^{N} x_j^{h_{i,j}}.
\]

Here \(x_i\) is the expression level of gene \(i\) and \(N\) is the number of genes in a genetic network. The non-negative parameters \(\alpha_i\) and \(\beta_i\) are rate constants that indicate the direction of mass flow. The real number exponents \(g_{i,j}\) and \(h_{i,j}\) are kinetic orders that reflect the intensity of interaction from gene \(j\) to \(i\). The above set of parameters defines an S-system model. To infer an S-system model is, thus, to estimate all of the \(2N (N + 1)\) parameters simultaneously.

As in a curve-fitting optimization problem, the goal here is to minimize the accumulated discrepancy between the gene expression data recorded in the data set (desired values) and the values produced by the inferred model (actual values). The desired values mean the real measurements in the laboratory experiments, whereas in our experiments they mean the data generated from the collected models. Thus, the performance of a certain model \(m\) can be defined directly as the mean squared error over the time period:

\[
p_m = \sum_{i=1}^{N} \sum_{t=1}^{T} \left( \frac{x_i^d(t) - x_i^a(t)}{x_i^a(t)} \right)^2,
\]

in which \(x_i^d(t)\) is the desired expression level of gene \(i\) at time \(t\), \(x_i^a(t)\) is the actual value obtained from the inferred model, \(N\) is the number of genes in the network and \(T\) is the number of time points in measuring gene expression data during the period. A small penalty term measuring the connection between genes can be added to the fitness function to discourage the connections (e.g. [11]), but the penalty is not used in this work because it is not the main focus here.

As can be observed, it is difficult to determine the large number of parameters involved in a GRN, especially when the complexity of regulation increases along with the number of genes involved. Although many intelligent computing techniques for parameter approximation have been proposed to derive solutions, approaches that are more efficient are still needed to resolve this high-dimensional problem. In this work, we establish a methodology for the construction of S-systems from two directions, bottom-up and top-down, to deal with the scalability problem. The former is to develop an enhanced GA–PSO hybrid approach to determine the system parameters defined above, and the latter is to tackle this problem in a divide-and-conquer manner that involves a network decomposition procedure to reduce the task complexity. The details are described in Sections 3.2 and 3.3, respectively.

3.2. An adaptive GA–PSO hybrid approach

To infer GRNs, our hybrid approach includes both GA and PSO procedures to exploit their respective advantages. As in other evolution-based methods, the first step in inferring a network model is to define an appropriate representation to characterize the solution. Here we take a direct encoding scheme to represent solutions for both GA and PSO parts, in which the network parameters described in Section 3.1 (i.e. \(\alpha_i, \beta_i, g_{i,j}\) and \(h_{i,j}\) in the above equation) are arranged as a linear string chromosome of floating-point numbers:

\[
\alpha_1 | g_{1,1} | g_{1,2} | \ldots | g_{1,n} | \beta_1 | h_{1,1} | h_{1,2} | \ldots | h_{1,n} \\
| \ldots | | \alpha_n | g_{n,1} | g_{n,2} | \ldots | g_{n,n} | \beta_n | h_{n,1} | h_{n,2} | \ldots | h_{n,n}
\]

With the above representation, the next step is to define a fitness function to evaluate the performance of each potential solution. As mentioned in the above section, the goal here is to minimize the accumulated discrepancy between the gene expression data recorded in the data set and the values produced by the inferred model. Therefore, the error function defined in Section 3.1 is used directly as the fitness function for performance measurement.
After defining the genetic representation and the fitness function, we are now to describe how the proposed approach operates. The main flow of our approach is illustrated in Fig. 1. Initially, a random population is generated and evaluated. Then the individuals are ranked according to their fitness, and separated into two parts. The first part includes elites (i.e. the best $p\%$ individuals of the entire population); they are preserved and enhanced by the PSO procedure and sent directly to a candidate list prepared for the next generation. The second part includes individuals with performance not as good as that of those in the first part (i.e. the worst $(1-p)\%$ individuals); therefore they are discarded. To replace the removed individuals, the same number of individuals is produced to form a parent pool, in which some individuals are generated randomly (i.e. $r\%$), and the remainder (i.e. $(1-p-r)\%$) are randomly selected from the ones already improved by the PSO procedure. Then this parent pool is used to create new individuals through the GA procedure, and the newly created individuals are sent to the candidate list. Once a new candidate list is formed, the individuals in this list are ranked again according to their fitness values and the new population is used for the next generation. The above procedure is repeated until the termination criterion is met.

3.2.1. The adaptive strategy
In our application, $p$ is designed to have a fixed value (estimated from a preliminary test), whereas $r$ is a variable whose value changes during the run to control the population diversity. In this work, the randomness rate $r$ increases linearly in proportion to the generation number to maintain the population diversity. This adaptive strategy is based on the observation that the PSO method converges very fast, but at the end of the run it tends to perform local search, and so a high rate of randomness is desired. Although more complicated strategies (for example, a non-linear function) can be used to change the randomness rate during the experimental run, a simple linear strategy described above is sufficient enough to deliver superior performance in our application of network inference. The corresponding effect will be shown in the experimental section below.

3.2.2. The PSO and GA parts
In the PSO procedure shown in Fig. 1, particles are potential solutions flying in the search space defined by the parameters, each of which has its position and velocity. To enhance the individual performance, the main operator here is the velocity updating for the particles, which combines the best position reached by the swarm of particles and the best position reached by a certain particle during its flying history. It has the effect that particles move toward the best position of the swarm. In this work, the velocity and the position of a particle at time step $t+1$ are updated from those at time step $t$ by the following rules modified from the original PSO [5]:

$$v_{id}^{t+1} = \chi(wv_{id}^{t} + c_1r_1(p_{id}^{t} - x_{id}^{t}) + c_2r_2(p_{gd}^{t} - x_{id}^{t})),$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1},$$

and $\chi = \frac{2}{2 - \sqrt{\phi^2 - 4\phi}}$.

In the above equations, $v_{id}, x_{id}$ are the velocity and position of particle $i$ at dimension $d$, $p_{id}$, and $p_{gd}$ are the previous

![FIGURE 1. The main flow of the proposed GA–PSO hybrid.](image-url)
best position of particle \( i \) and the best position of the swarm, respectively, and \( w \) is the inertia weight. It controls the momentum of the particle by weighting the contribution of the particle’s previous velocity (i.e. the influence of the memory). The coefficients \( c_1 \) and \( c_2 \) are two positive acceleration constants used to scale the contribution of the cognitive and social components; they are often determined empirically. The values \( r_1 \) and \( r_2 \) are both random values within the range \([0, 1]\). The products \( c_1 r_1 \) and \( c_2 r_2 \) thus stochastically control the overall velocity of a particle. In addition, \( \chi \) is the constriction factor originally introduced in [5] to constrict the velocities of particles to achieve the balance of exploration–exploitation during the search, and to guarantee that the particles converge to a stable point, and \( \phi \) is a parameter often used to control the convergence characteristics of the PSO.

As mentioned, the GA part is used to create new individuals to replace the ones discarded. In this procedure, the tournament selection strategy is employed to choose parent pairs. For the selected particles old1 and old2, the following crossover operator is implemented as below to create two new particles new1 and new2:

\[
\text{new}_1(x_{id}) = (\text{old}_1(x_{id}) + \text{old}_2(x_{id}))/2 - \varphi_1 \times \text{old}_2(v_{id}), \\
\text{new}_2(x_{id}) = (\text{old}_1(x_{id}) + \text{old}_2(x_{id}))/2 - \varphi_2 \times \text{old}_1(v_{id}).
\]

In this operator, \( \varphi_1 \) and \( \varphi_2 \) are uniform random variables with values between 0 and 1. If the position of the newly created particle falls outside the specified range, it is set to the maximal value \( x_{id\_max} \). As can be observed, this operator mainly serves to incorporate the velocities of the two parents. It produces two children with positions between the two parents but accelerated away from the current directions in order to increase population diversity. Then the commonly used non-uniform mutation procedure is performed to fine-tune the numerical values of the individuals in order to infer the network parameters.

### 3.2.3 Model selection

As analyzed previously, inferring a gene network is to determine network parameters for the chosen computational network model, and an adaptive GA–PSO method is developed to search the space constituted by these parameters to find the appropriate solutions. However, due to the cost consideration of laboratory experiments for data collection, in the real-world situations, the number of data points available is often smaller than that of the parameters to be determined; that is, the network inference is in fact an under-determined problem. It is thus possible to obtain various combinations of network parameters (i.e. solutions) with error very close to zero. From the point of view of numerical optimization, these combinations can all be considered as feasible solutions. To infer solutions with special biological meaning, domain knowledge must be integrated into the computational methods. A simple and efficient way is to take the form of parameter constraint for knowledge representation.

This way, a biologist can explicitly encode his prior knowledge of the gene concentrations or the regulation relationships between genes in the forms of numerical constraints (e.g. \( c_1 \leq np_i \leq c_2 \) or \( np_j \leq np_i \), in which \( c_1, c_2 \) are constants and \( np_i, np_j \) are network parameters). The constraints specified for parameters can then be used to restrain the solution search, and only the ones meaningful to the domain experts will be derived.

### 3.3 Gene clustering

For small networks, the search-based GA–PSO approach presented in the above section can be used to infer the entire network models directly. Additionally, to solve more complicated reconstruction tasks, we also develop a clustering-based method to decompose the search space and to reduce the task complexity. Clustering is a useful exploratory technique for the analysis of gene expression data. The hypothesis of gene clustering is that the genes in a cluster may share some common functions or regulatory elements and can thus be considered and modeled together. In our method, a clustering technique is first employed to group the genes into tightly coupled small-scale networks, based on the analysis of their corresponding expression data, and then the small networks can be decomposed again in a similar way until the resulting networks can be directly modeled. Then the small networks are directly inferred from the expression data. Once all the small networks have been obtained, they can be regarded as self-contained components of the original system, and assembled together manually or by the given learning algorithm.

In our current implementation, the self-organization feature map (SOM, [15]) network is adopted for gene clustering, as it provides better inter-cluster and intra-cluster distance measurements (i.e. separation and compactness) than other widely used clustering methods (i.e. \( k \)-means and hierarchical clustering) in our preliminary tests on the modeling cases here. Before a clustering method is applied to the expression data, some features of the data set have to be identified so that the clustering method can find the potentially meaningful relationships between the data. As there are no predefined data features to be selected in gene expression profiles, a feature extraction procedure needs to be performed. Here we use the wavelet transform (WT) technique to extract data features from the waveforms derived from the gene expression data of different time points. Figure 2 depicts the overall procedure of our gene clustering.

The WT theory has been widely used in many signal-processing applications, e.g. [16]. It decomposes a signal into a set of basis building blocks called wavelets. These building blocks are actually a family of functions derived from a single generating function (i.e. the mother wavelet) by scaling (dilation/compression) and translation operations. In its original form [17], the continuous WT (CWT) is defined as the
convolution of a time series \( x(t) \) with a wavelet function \( \psi(t) \):

\[
CWT(\psi) = \frac{1}{a^{1/2}} \int_{-\infty}^{\infty} x(t) \cdot \psi^*(\frac{t-b}{a}) dt.
\]

Where \( x(t) \) is the signal to be analyzed, \( a \) and \( b \) represent the scaling factor and the translation along the time axis, respectively (to scale the wavelet at time \( b \) with a factor of \( a \)) and the superscript asterisk denotes the complex conjugation of \( \psi(t) \).

In the CWT form, the scaling and translation factors \( a \) and \( b \) change continuously. For practical computation, the discrete wavelet transfer (DWT) is often used, and which can be achieved by restricting the variation in scale and translation (usually to powers of 2). When the scale is changed in powers of 2, that is, \( a = 2^m \) and \( b = n2^m \) (\( n, m \) are integers), the DWT can thus be described as

\[
DWT_{\psi}(m, n) = 2^{-m/2} \int_{-\infty}^{\infty} x(t) \cdot \psi^*(2^m t - n) dt.
\]

For discrete-time signals, the above wavelet computation can be performed equivalently by simply using the filtering processes. That is, the multi-resolution analysis is carried out by using two-channel filter banks composed of a low-pass filter \( h \) and its complementary high-pass filter \( g \). With the specific initial condition and standard quadrature mirror filter condition for \( h \), a sequence of filters with increasing length can be derived. Figure 3 illustrates the sub-band decomposition of DWT implementation for a signal \( x \). Each stage in the figure consists of two filters and two downsamplers by 2 (denoted as \( \downarrow 2 \)). The scaling and wavelet functions can be obtained from the filters. Then the approximation coefficient \( a_l \) and detailed coefficient \( d_i \) of the DWT decomposition at each resolution \( i \) can be calculated from these functions accordingly. In this way, the original signal \( x \) can be decomposed and described as

\[
x = a_l + \sum_{i=1}^{l} d_i,
\]

where \( l \) is the resolution level the signal to be analyzed. More details on the formulation can be found in [17].

As can be seen, DWT can compress an original signal consisting of many data points into a few parameters (i.e. wavelet coefficients) that characterize the behavior of the signal. These coefficients provide a compact representation that shows the energy distribution of the signal in time and frequency. Therefore, in this work we collect the wavelet coefficients derived from the time-varying gene regulatory signals as features of the signals. Figure 4 shows the typical result of a DWT for a certain gene (produced by MATLAB Wavelet toolbox with four levels multi-resolution analysis), in which \( s \) is the original gene expression data, \( a_4 \) is the approximation coefficient and \( d_1-d_4 \) are the detailed coefficients (in this case, \( s = a_4 + d_4 + d_3 + d_2 + d_1 \)). The high-pass wavelet coefficients are then collected and used to form the input vector of the SOM network for gene clustering.
The SOM network is composed of an input layer and an output layer, in which the input layer is fully connected to the output layer as shown in Fig. 2. The input to the network is a feature vector (i.e. the wavelet coefficients here) with \( n \) dimensions, represented as \( \mathbf{X} = [x_1, x_2, \ldots, x_n]^T \). Each neuron \( j \) on the map (i.e. the output layer) is represented by an \( n \)-dimensional weight vector \( \mathbf{W}_j = [w_{j1}, w_{j2}, \ldots, w_{jn}]^T \) \( (j = 1 \sim N, N \) is the number of neurons) which is initialized randomly. The main function of such networks is to map the input data from an \( n \)-dimensional space to a much lower (e.g. two) dimensional space, while maintaining the original topological relations between the input data.

When an input vector is presented to the network, the connected relation between the input vector \( \mathbf{X} \) and each weight vector \( \mathbf{W}_j \) is calculated according to the dissimilarity measure (e.g. Euclidean distance between the vectors) as \( \| \mathbf{X} - \mathbf{W}_j \| \). The winning neuron \( k \) is then identified as the one with the best match or the nearest to \( \mathbf{X} \):

\[
\| \mathbf{X} - \mathbf{W}_k \| = \min_j \{ \| \mathbf{X} - \mathbf{W}_j \| \}.
\]

Then the weight vectors of the winning neuron and its neighborhood neurons are updated by the following rules:

\[
\begin{align*}
\mathbf{W}_i(t+1) &= \mathbf{W}_i(t) + \alpha(t)(\mathbf{X}(t) - \mathbf{W}_i(t)), & \text{if } i \in N_k(t), \\
\mathbf{W}_i(t+1) &= \mathbf{W}_i(t), & \text{if } i \notin N_k(t),
\end{align*}
\]

where \( \alpha(t) \) and \( N_k(t) \) are the learning rate and the neighborhood function, respectively. They have prespecified values and gradually decrease to zero with time \( t \). With the above adaptation procedure, the weight vectors will converge to stable values. As a result, the SOM network can form gene clusters of the input vectors (expression data) with similar properties by groups of neurons with similar weights.

4. EXPERIMENTS AND RESULTS

After describing the proposed methodology for network reconstruction, we present three series of experiments for performance evaluation. The first series compares the performance of different inference methods, in which five different data sets for small-scale networks were used. The second series of experiments is to examine whether our method can be used to infer robust networks. Finally, the proposed inference method was coupled with a clustering technique to reconstruct complicated networks with more gene nodes.

4.1. Performance evaluation for inferring networks

To evaluate our hybrid GA–PSO method objectively, we collect several data sets that are commonly used and are available...
from the related works, to conduct the experiments on network reconstruction. In addition, to compare the proposed method with others, we have implemented different methods, including the ordinary GA and PSO, and two hybrid GA–PSO methods (which have been widely used for performance comparison [8, 9]).

In the experiments, the first data set was derived from a five-node regulatory network as reported in [18]. To collect time-series data, we started the network operations and continued the operations for 30 simulation time steps. The relationships among the nodes are described as follows:

\[
\begin{align*}
\dot{X}_1 &= 15.0X_3X_5^{-0.1} - 10.0X_2^{1.0}, \\
\dot{X}_2 &= 10.0X_1^{1.0} - 10.0X_2^{2.0}, \\
\dot{X}_3 &= 10.0X_2^{-0.1} - 10.0X_2^{-1}X_3^{2.0}, \\
\dot{X}_4 &= 8.0X_1^{2.0}X_5^{-1} - 10.0X_4^{2.0}, \\
\dot{X}_5 &= 10.0X_4^{2.0} - 10.0X_5^{2.0}.
\end{align*}
\]

The second data set used in the experiment was collected from [19]. It is a data set slightly modified from the first data set (i.e. the coefficient of the first equation was changed from 15.0 to 5.0). The third data set is a 10-node network described in [13], which is as follows:

\[
\begin{align*}
\dot{X}_1 &= 5.0X_4X_9^{2.0} - 10.0X_7^{2.0}, \\
\dot{X}_2 &= 10.0X_3X_8^{1.0} - 10.0X_2^{2.0}, \\
\dot{X}_3 &= 8.0X_1^{1.0}X_4^{1.0} - 10.0X_3^{2.0}, \\
\dot{X}_4 &= 10.0X_9^{2.0}X_9 - 10.0X_4^{2.0}, \\
\dot{X}_5 &= 10.0X_2^{2.0}X_6^{-1.0} - 10.0X_2^{2.0}, \\
\dot{X}_6 &= 5.0X_9^{2.0}X_9^{1.0} - 10.0X_6^{2.0}, \\
\dot{X}_7 &= 10.0X_6X_8^{-1.0} - 10.0X_7^{2.0}, \\
\dot{X}_8 &= 5.0X_1X_2^{2.0}X_7 - 10.0X_8^{2.0}, \\
\dot{X}_9 &= 10.0X_3X_8^{2.0} - 10.0X_9^{2.0}, \\
\dot{X}_{10} &= 8.0X_1^{2.0}X_7^{-1.0} - 10.0X_2^{2.0}.
\end{align*}
\]

In addition to the above artificial data sets, two laboratory data sets were taken from [20, 21]. The experimental materials and methods for data collection can be found in their original works. As in the artificial data sets, the expression data (generated from the models recorded in [20, 21]) were the only available information for network inference, and therefore we used the same way of comparing the behaviors of the original and inferred networks to validate the experiments. Here, one data set is a five-node network consisting of two genes and the proteins they produced in the regulatory process. The other is a four-node network that describes a fermentation process using high-ethanol-tolerance yeast to produce ethanol. The two data sets are described below:

\[
\begin{align*}
\dot{X}_1 &= 5X_3X_5^{-1}X_6 - 10X_7^{2}, \\
\dot{X}_2 &= 10X_1^{1}X_7 - 10X_2^{2}, \\
\dot{X}_3 &= 10X_2^{-1}X_8 - 10X_2^{-1}X_3^{2}, \\
\dot{X}_4 &= 8X_2^{2}X_5^{-1}X_6 - 10X_2^{2}, \\
\dot{X}_5 &= 10X_4^{2}X_7 - 10X_5^{2}, \\
\dot{X}_6 &= 3.1125X_1^{0.8993}X_2^{0.2771} - 0.4777X_1^{1.6345}X_2^{-0.2768}, \\
\dot{X}_7 &= 3.3067 - 1.8574X_5^{0.5929}X_2^{0.2960}, \\
\dot{X}_8 &= 0.5433X_1^{0.8782}X_2^{0.2282} - 0.0735X_3^{0.8352}, \\
\dot{X}_9 &= 0.3568X_1^{0.6600}X_2^{3.3187}.
\end{align*}
\]

FIGURE 7. The original (left) and inferred (right) network behaviors for data set 3.
FIGURE 8. The original (left) and inferred (right) network behaviors for data set 4.

FIGURE 9. The original (left) and inferred (right) network behaviors for data set 5.

TABLE 1. Results obtained by different methods for data set 1.

<table>
<thead>
<tr>
<th></th>
<th>GA</th>
<th>PSO</th>
<th>H1 [8]</th>
<th>H2 [9]</th>
<th>r = 0.1</th>
<th>r = 0.3</th>
<th>r = 0.5</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg</td>
<td>1.3075</td>
<td>0.6524</td>
<td>0.6150</td>
<td>0.0487</td>
<td>0.0370</td>
<td>0.0311</td>
<td>0.0450</td>
<td>0.0271</td>
</tr>
<tr>
<td>Best</td>
<td>0.6498</td>
<td>0.4365</td>
<td>0.3471</td>
<td>0.0192</td>
<td>0.0083</td>
<td>0.0106</td>
<td>0.0183</td>
<td>0.0077</td>
</tr>
<tr>
<td>Worst</td>
<td>2.0916</td>
<td>0.9091</td>
<td>0.7729</td>
<td>0.0870</td>
<td>0.0521</td>
<td>0.0473</td>
<td>0.0668</td>
<td>0.0512</td>
</tr>
<tr>
<td>SD</td>
<td>0.5107</td>
<td>0.1554</td>
<td>0.1498</td>
<td>0.0217</td>
<td>0.0170</td>
<td>0.0132</td>
<td>0.0165</td>
<td>0.0148</td>
</tr>
</tbody>
</table>

TABLE 2. Results obtained by different methods for data set 2.

<table>
<thead>
<tr>
<th></th>
<th>GA</th>
<th>PSO</th>
<th>H1 [8]</th>
<th>H2 [9]</th>
<th>r = 0.1</th>
<th>r = 0.3</th>
<th>r = 0.5</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg</td>
<td>0.4210</td>
<td>0.4982</td>
<td>0.1174</td>
<td>0.0246</td>
<td>0.0242</td>
<td>0.0180</td>
<td>0.0169</td>
<td>0.0164</td>
</tr>
<tr>
<td>Best</td>
<td>0.2322</td>
<td>0.2370</td>
<td>0.0792</td>
<td>0.0177</td>
<td>0.0110</td>
<td>0.0091</td>
<td>0.0091</td>
<td>0.0093</td>
</tr>
<tr>
<td>Worst</td>
<td>0.6494</td>
<td>0.6603</td>
<td>0.1689</td>
<td>0.0318</td>
<td>0.0323</td>
<td>0.0265</td>
<td>0.0239</td>
<td>0.0218</td>
</tr>
<tr>
<td>SD</td>
<td>0.1506</td>
<td>0.1564</td>
<td>0.0373</td>
<td>0.0051</td>
<td>0.0077</td>
<td>0.0067</td>
<td>0.0044</td>
<td>0.0039</td>
</tr>
</tbody>
</table>
TABLE 3. Results obtained by different methods for data set 3.

<table>
<thead>
<tr>
<th></th>
<th>GA</th>
<th>PSO</th>
<th>H1 [8]</th>
<th>H2 [9]</th>
<th>(r = 0.1)</th>
<th>(r = 0.3)</th>
<th>(r = 0.5)</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>This work</td>
<td>4.6015</td>
<td>4.2768</td>
<td>3.5788</td>
<td>0.8861</td>
<td>0.4662</td>
<td>0.4776</td>
<td>0.3731</td>
<td>0.3796</td>
</tr>
<tr>
<td>Avg</td>
<td>4.6692</td>
<td>3.6644</td>
<td>2.6335</td>
<td>0.5395</td>
<td>0.1560</td>
<td>0.3251</td>
<td>0.1306</td>
<td>0.2597</td>
</tr>
<tr>
<td>Best</td>
<td>6.2123</td>
<td>5.0123</td>
<td>4.5541</td>
<td>1.1289</td>
<td>0.6432</td>
<td>0.6644</td>
<td>0.4963</td>
<td>0.5132</td>
</tr>
<tr>
<td>Worst</td>
<td>0.8249</td>
<td>0.5064</td>
<td>0.7093</td>
<td>0.1996</td>
<td>0.1482</td>
<td>0.1159</td>
<td>0.0996</td>
<td>0.0898</td>
</tr>
<tr>
<td>SD</td>
<td>0.8249</td>
<td>0.5064</td>
<td>0.7093</td>
<td>0.1996</td>
<td>0.1482</td>
<td>0.1159</td>
<td>0.0996</td>
<td>0.0898</td>
</tr>
</tbody>
</table>

TABLE 4. Results obtained by different methods for data set 4.

<table>
<thead>
<tr>
<th></th>
<th>GA</th>
<th>PSO</th>
<th>H1 [8]</th>
<th>H2 [9]</th>
<th>(r = 0.1)</th>
<th>(r = 0.3)</th>
<th>(r = 0.5)</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>This work</td>
<td>0.1608</td>
<td>0.0336</td>
<td>0.0486</td>
<td>0.0189</td>
<td>0.0018</td>
<td>0.0017</td>
<td>0.0017</td>
<td>0.0016</td>
</tr>
<tr>
<td>Avg</td>
<td>0.0191</td>
<td>0.0105</td>
<td>0.0145</td>
<td>0.0077</td>
<td>0.0011</td>
<td>0.0009</td>
<td>0.0006</td>
<td>0.0008</td>
</tr>
<tr>
<td>Best</td>
<td>0.4589</td>
<td>0.0705</td>
<td>0.0728</td>
<td>0.0381</td>
<td>0.0023</td>
<td>0.0028</td>
<td>0.0036</td>
<td>0.0028</td>
</tr>
<tr>
<td>Worst</td>
<td>0.1527</td>
<td>0.0248</td>
<td>0.0204</td>
<td>0.0121</td>
<td>0.0006</td>
<td>0.0004</td>
<td>0.0006</td>
<td>0.0006</td>
</tr>
<tr>
<td>SD</td>
<td>0.1527</td>
<td>0.0248</td>
<td>0.0204</td>
<td>0.0121</td>
<td>0.0006</td>
<td>0.0004</td>
<td>0.0006</td>
<td>0.0006</td>
</tr>
</tbody>
</table>

TABLE 5. Results obtained by different methods for data set 5.

<table>
<thead>
<tr>
<th></th>
<th>GA</th>
<th>PSO</th>
<th>H1 [8]</th>
<th>H2 [9]</th>
<th>(r = 0.1)</th>
<th>(r = 0.3)</th>
<th>(r = 0.5)</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>This work</td>
<td>0.6061</td>
<td>0.1595</td>
<td>0.2096</td>
<td>0.0993</td>
<td>0.0558</td>
<td>0.0479</td>
<td>0.0646</td>
<td>0.0463</td>
</tr>
<tr>
<td>Avg</td>
<td>0.1358</td>
<td>0.1021</td>
<td>0.0927</td>
<td>0.0703</td>
<td>0.0383</td>
<td>0.0348</td>
<td>0.0282</td>
<td>0.0334</td>
</tr>
<tr>
<td>Best</td>
<td>1.1896</td>
<td>0.3536</td>
<td>0.2861</td>
<td>0.1453</td>
<td>0.0743</td>
<td>0.0733</td>
<td>0.0895</td>
<td>0.0718</td>
</tr>
<tr>
<td>Worst</td>
<td>0.3886</td>
<td>0.0797</td>
<td>0.0691</td>
<td>0.0273</td>
<td>0.0113</td>
<td>0.0130</td>
<td>0.0229</td>
<td>0.0128</td>
</tr>
<tr>
<td>SD</td>
<td>0.3886</td>
<td>0.0797</td>
<td>0.0691</td>
<td>0.0273</td>
<td>0.0113</td>
<td>0.0130</td>
<td>0.0229</td>
<td>0.0128</td>
</tr>
</tbody>
</table>

FIGURE 10. Comparison of the fitness curves by five different methods for data set 1.

increase the randomness rate (i.e. \(r\) varies from 10 to 50%) in proportion to the generation number, to maintain population diversity.

Twenty independent runs were conducted and the population size was 400 for each run. Figures 5–9 compare the behaviors of the original networks and the networks inferred by our approach, in which the \(x\)-axis represents time step, and the \(y\)-axis, the concentration of different gene components. As can be seen, all the network models can be inferred successfully: the behaviors
of the inferred networks are very similar to the original ones, in which many of the genes generate almost identical data sequences.

As mentioned above, the other four methods were also implemented and used to reconstruct the network reversely from the same data sets. For each method, 20 independent runs were conducted for each data set, and the experimental settings remained the same as in the runs by the proposed method. Tables 1–5 show the results of the five data sets, respectively, in which the mean, standard deviation, best and worst performance of the runs are listed for each method. From these tables, we can observe that the proposed method, with either the fixed or the dynamic strategy, outperforms all others in inferring gene networks. Our hybrid method also gives smaller standard deviations for all sets of experimental runs than others. It indicates that our method is more stable than others. To observe the converged system behaviors of different computational methods, Figs 10–14 compare the fitness curves of the best individuals during the runs (values are averaged from all runs for each data set) in inferring networks. These results clearly show that compared with others, the proposed methods perform better.
TABLE 6. Results of inferring networks with random noises.

<table>
<thead>
<tr>
<th>Data set 1</th>
<th>3% noise</th>
<th>5% noise</th>
<th>10% noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>fit1</td>
<td>0.0640 (0.0149)</td>
<td>0.1528 (0.0148)</td>
<td>0.3737 (0.0145)</td>
</tr>
<tr>
<td>fit2</td>
<td>0.0596 (0.0279)</td>
<td>0.0667 (0.0185)</td>
<td>0.2262 (0.1359)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data set 2</th>
<th>3% noise</th>
<th>5% noise</th>
<th>10% noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>fit1</td>
<td>0.0518 (0.0054)</td>
<td>0.1217 (0.0124)</td>
<td>0.4274 (0.0073)</td>
</tr>
<tr>
<td>fit2</td>
<td>0.0534 (0.0187)</td>
<td>0.1045 (0.0464)</td>
<td>0.2763 (0.0922)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data set 3</th>
<th>3% noise</th>
<th>5% noise</th>
<th>10% noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>fit1</td>
<td>0.5433 (0.1318)</td>
<td>0.6069 (0.1188)</td>
<td>1.4378 (0.1586)</td>
</tr>
<tr>
<td>fit2</td>
<td>0.8785 (0.2964)</td>
<td>1.1761 (0.4135)</td>
<td>2.4701 (0.8951)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data set 4</th>
<th>3% noise</th>
<th>5% noise</th>
<th>10% noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>fit1</td>
<td>0.0439 (0.0004)</td>
<td>0.1127 (0.0011)</td>
<td>0.4235 (0.0011)</td>
</tr>
<tr>
<td>fit2</td>
<td>0.0108 (0.0035)</td>
<td>0.0585 (0.0334)</td>
<td>0.2005 (0.1479)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data set 5</th>
<th>3% noise</th>
<th>5% noise</th>
<th>10% noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>fit1</td>
<td>0.0889 (0.0318)</td>
<td>0.1284 (0.0346)</td>
<td>0.1834 (0.0540)</td>
</tr>
<tr>
<td>fit2</td>
<td>0.1414 (0.0358)</td>
<td>0.1715 (0.0641)</td>
<td>0.4084 (0.2967)</td>
</tr>
</tbody>
</table>

The method can obtain better solutions within shorter inferring time. It is more efficient than and is superior to all other methods of network inference.

After evaluating the proposed approach, we took an example created from the popular gene network simulation software Genexp [22], to conduct a series of experiments to illustrate the effect of the user-specified constraints mentioned in Section 3.2.2. The network has eight nodes and therefore there are 144 parameters to be determined in total. In the experiments, we first produced 20 independent runs without constraints to infer the network. Then we conducted another two sets of experiments (each set included 20 successful runs) with constraints for comparison. One is to arbitrarily choose 72 of 144 network parameters and specify the value range for each of them; and the other, to specify ranges for all 144 parameters. To observe the correlation of the solutions inferred from the runs, we project the solutions into a SOM. The results are presented in Fig. 15 (two cases with different parameter constraints), in which solutions marked with labels A, B and C represent the results obtained from the runs without constraints, with 72 constraints and with 144 constraints, respectively. As can be seen, more constraints cluster the solutions together within a...
Adaptive GA—PSO Approach with Gene Clustering

FIGURE 18. The behaviors of the original (a) and inferred (b) networks.

relatively smaller region (e.g. label C). They indicate that if the user has some domain knowledge and uses constraints to convey his knowledge, more specific models can be derived as expected by the proposed approach.

4.2. Network robustness

In addition to the issue of parameter fitting, network robustness is an important factor to be considered in network modeling. In the process of inferring GRNs from the measured gene expression data set, the target networks may be perfectly inferred. But it should be noted that the data samples collected at different time points from the microarray experiments are very limited, and which often causes the over-fitting problem; that is, the models perfectly fitting the data set generally give poor performance in prediction. In addition, as is known, gene expression measurements are unreliable; they always contain a certain amount of noises. Under such circumstances, the obtained networks are not as robust as expected.

To infer a robust gene network, the criterion of adding noises to the data set for network inference is proposed [23]. In fact, it has been shown that adding noises of small amplitude to the data set is equivalent to a Tikhonov regulariztion [24]. It is also known that the Tikhonov regularizers are especially suitable for modeling work with insufficient sampling data. Therefore, when a regularization term is hard to find in the modeling process, adding noises has been advocated as a simple alternative way for regularization. In this way, the prediction capability of the inferred network on the data set is not perfect any more, whereas the robustness will be optimal and the solution stable [24].

In this section, we adopt the above strategy to infer robust networks. Instead of expanding the original measured data set with a set of noisy duplicates (to overcome the problem of insufficient training samples, as reported in [23]), here we introduced some random noises to the desired data to simulate the effect of data disturbance. Three sets of experiments have been conducted for comparison, in which the original five data sets reported in Section 4.1 randomly added/deleted a small amount of noises up to 3% (i.e. +3 to −3% uniformly), 5% and 10%, respectively. Table 6 presents the results of inferring networks with different levels of random noises. In this table, fit1 reports the fitness (i.e. the accumulated mean squared error) in inferring a model to fit the disturbed data set, and fit2 indicates the discrepancy (in terms of the mean squared error as above) between the inferred model and the original data set. Each entry in the table is the average obtained from 10 independent runs (with the standard deviation in the parentheses). As can be expected, when the noise was injected and the amount was increased gradually, the random effect became more difficult to capture. Though the model became difficult to reconstruct perfectly in this way, however, the proposed approach still can effectively preserve the global system behavior (with relatively small fitness/error), and at the same time the robustness of an inferred network can be enhanced.

<table>
<thead>
<tr>
<th>TABLE 7. The details of each subgroup.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Group (#genes)</strong></td>
</tr>
<tr>
<td>sub-a (8)</td>
</tr>
<tr>
<td>sub-b (5)</td>
</tr>
<tr>
<td>sub-c (4)</td>
</tr>
<tr>
<td>sub-d (5)</td>
</tr>
</tbody>
</table>

The Computer Journal, Vol. 54 No. 9, 2011
4.3. Performance of gene clustering

In the inference of gene networks, when the number of genes in a network increases in respect to the increasing functional complexity of the network, the number of network parameters increases rapidly and makes the search difficult. To model large systems with more genes, it is thus important to first...
construct a coarse-grained description of the system. This section demonstrates how the gene clustering method can help to infer coarse-grained network models from data.

Two data sets have been used in the experiments of modeling large networks. The first data set is a 10-node artificial data set created manually by Genexp. To collect data, initial parameters were specified for network operation and then the expression data were recorded for 30 consecutive time points. Since it was difficult to reconstruct the entire network from these time-series data directly by the inference method, the gene clustering method described in Section 3.3, including the procedures of WT and the SOM, was used to group the genes. Two clusters were observed, one for genes 1–3, the other for genes 5–9. Obviously, genes 4 and 10 did not belong to any of the above two clusters. After measuring the gene distance and calculating the Pearson correlation coefficients between the genes, we decided to organize the genes into four parts as shown in Fig. 16, in which genes 4 and 10 were considered as outliers. In the experiments, the proposed GA–PSO approach was used to infer networks for the two major clusters, and then the network parameters obtained were used as initial settings to infer the overall network at a higher level. The results for the two sub-nets are shown in Fig. 17a and b, respectively, and Fig. 18 presents the system behaviors of the original and reconstructed networks over the 10 genes. It can be observed that the behavior of the trained network is very similar to the original one, in which many of them have almost identical data sequences. These results evidently show the efficiency of the proposed approach.

The second data set used in this set of experiments is the real experimental data set Rat central nervous system (CNS), taken from [25]. This data set includes expression data of 112 genes collected from nine time points of different phases (embryonic, postnatal and adult). Again, the gene clustering method was used to group the genes. Out of the data from the 112 genes, 103 of them were categorized into six different clusters and 9 genes did not belong to any cluster. One of our clusters, consisting of 19 genes, is very similar to one reported in a previous study dealing with rat CNS data [25] (containing the 17 gene cluster, in fact). To be consistent with the previous study and to preserve the meaning of the cluster as in the original work, we decided to use the 17 gene cluster reported in [25] as the target network to be reconstructed.

As the genes within the same cluster are closely related, it is not practical to group them by the same clustering method again. Therefore, once the above target network had been determined, the genes were decomposed into four subgroups according to the mutual information between them, and by which some genes belonged to more than one subgroup. Table 7 lists the details of the decomposed results in which the four subgroups have eight, five, four and five nodes, respectively. Then the four subnets, and afterward the target network, were built. Figure 19a–d presents four sets of behaviors of the original and inferred sub-networks group by group. Again, very similar behaviors between the original and inferred sub-networks can be observed. After that, the network parameters of the inferred sub-nets were used as the initial settings for the individuals in the experiments of inferring the overall network. Figure 20 shows the corresponding network behaviors. It indicates that the proposed gene clustering approach can be efficiently and successfully used to model networks of relatively large size (Table 7).

5. CONCLUSIONS

In this work, we have emphasized the importance of inferring GRNs from gene expression profiles, and indicated that the two most crucial steps in network inference are selecting the network model and fitting the time-series data into the network parameters. Depending on the biological level to be studied, many models have been proposed to simulate GRNs, and different computational methods have also been developed to reconstruct networks accordingly. Instead of identifying which model and method are most suitable for network reconstruction, here we have concentrated on establishing a practical methodology that can model GRNs and is scalable for inferring large-scale networks. As S-systems can work as dynamical systems as GRNs do, our work has adopted this model to represent GRNs, and developed a hybrid GA–PSO method to infer network models from expression data.
results have shown that our method can infer networks successfully and has prominent performance. Meanwhile our work takes the form of knowledge-based constraints to restrain the proposed search-based method and can evolve meaningful solutions. The importance of network robustness has also been pointed out, and the strategy of adding noises to the original data has been applied to inferring robust networks.

To deal with the scalability problem, we have developed a gene clustering method to perform network decomposition. As a new application, our method combines the well-known DWT and SOM techniques to exploit their advantages in signal decomposition and data grouping, respectively. The features of gene expression data are extracted through the DWT procedure and collected to form the input vectors of a SOM network. With this combined method, gene clusters can thus be obtained. To verify the presented method, experiments have been conducted to demonstrate how it can be used to infer large networks progressively and hierarchically.

On the basis of the work presented, we are currently developing new gene clustering methods that can consider more characteristics of gene regulation simultaneously in feature extraction. The new methods are expected to be particularly suitable for GRN modeling. We are also conducting more extensive experiments to investigate the effect of using user-specified numerical constraints on genes to restrain the solution search. In addition, we plan to develop advanced methods that are able to automatically extract more useful domain information from various biological knowledge bases and to integrate the information into the search-based inference methods for network reconstruction.

FUNDING

This work has been supported by the National Science Council of Taiwan, under contract NSC-97-2221-E-110-063-MY2.

REFERENCES