3

GNGS: An Artificial Intelligent Tool for Generating and Analyzing Gene Networks from Microarray Data

Austin H. Chen[1] and Ching-Heng Lin[2]
[1]*Department of Medical Informatics, Tzu-Chi University*
[1,2]*Graduate Institute of Medical Informatics, Tzu-Chi University*
Taiwan

1. Introduction

The completion of the Human Genome Project has been recognized as a great achievement in the study of biomedicine; the project not only provides information regarding human genes but also provides new ways to study human diseases such as cancers. High-throughput techniques, such as microarray experiments, have emerged as a method of study that measures the level of gene expression in gene networks. Since microarray experiments can produce thousands of datasets under various experimental conditions simultaneously, it is now feasible to study gene interactions and regulatory networks. How to analyze and interpret the results of these analyses, however, has become an important research area in bioinformatics.

In the study of biological cellular behavior, understanding how biological activities are governed by the relationships among genes, RNA, and proteins is a common challenge. Gene networks represent such connectivity. A gene network consists of a group of genes that interact among themselves in order to synthesize proteins. Recently, genome-wide gene expression microarray data relevant to the yeast cell cycle has been collected (Spellman et al., 1998; Cho et al., 1998; Zhu et al., 2000). Since the gene expression profile data is a record of the network interactions between the regulators and the target genes, it is possible to use this information to trace these complex relationships. A variety of computer clustering methods have been developed in order to group together genes with similar patterns of expression (Eisen et al., 1998; Tamayo et al., 1999; Tavazoie et al, 1999).

Previous efforts at modeling gene networks from high dimensional datasets have generally fallen into one of three classes, either employing Boolean networks (D'haeseleer et al., 1999; Husmeier et al., 2005), which are restricted to logical relationships between variables, or using systems of differential equations (Chen et al., 1999; Sakamoto & Iba, 2001; Thomas, 1990) to model the continuous dynamics of coupled biological reactions. The work of Friedman et al. (2000) uses Bayesian networks to analyze expression data. The statistical framework of Bayesian learning, since it deals with uncertainly, is designed for domains

with a large number of variables and for handling noisy data. Another advantage of this probabilistic approach is the ability to combine prior knowledge with the information extracted from data.

A Bayesian network is a graphical model that finds probabilistic relationships among variables (i.e. genes) of the system. Bayesian networks are popular decision support models (Cooper & Herskovits, 1992; Husmeier, 2005) because they inherently model the uncertainty in the data. In addition, Bayesian networks successfully amalgamate probability theory and graph theory to efficiently model multidimensional probability distributions by searching for independent relationships in the data (Gevaert et al., 2006; Heckerman, 1995). Other features that make Bayesian networks attractive candidates for modeling gene expression data include the ability to handle noisy or missing information, handle hidden variables, and make causal inferences. (Beal et al, 2005)

Currently, a user-friendly system that can display and analyze various gene networks from microarray experimental datasets is urgently needed. In this study, our goal is to develop a gene network generating system (GNGS) that can generate the gene networks of the yeast cell cycle from experimental microarray data as well as analyze the performance of gene networks using five different Bayesian network algorithms.

2. Methods

In this study, three kinds of datasets were used. The first two datasets are Alarm (Beinlich, 1989) and Asia (Lauritzen & Spiegelhalter, 1988) networks. These two datasets were commonly used in Bayesian networks, and the known structure of the Alarm and Asia networks are used to compare the performance of different Bayesian network algorithms.

The third dataset used in this study is S. cerevisiae cell cycle gene expression data collected by Spellman et al. (1998). This dataset contains four medium time series: 18, 24, 17 and 14 time series points for alpha, cdc15, cdc28 and elu respectively. In the assessment of a gene network, we use each of the three medium time series: alpha, cdc15, and cdc28.

After normalizing the gene expression data, we sorted these values into three classes based on Friedman's threshold value of 0.5 (Friedman et al., 2000). The data was then translated into 3 discrete values:

Data representation
- over-expressed = +1
- normal expressed = 0
- under-expressed = -1

The results were compared with a known YPL256C sub network (Dejori, 2002). In order to compare the performance of gene networks generated from different Bayesian network algorithms (Kim et al., 2004), we defined specificity and sensitivity as Formula 1 and Formula 2.

$$\text{Sensitivity} = \frac{\#\ \text{correctly estimated edges}}{\#\ \text{edges in the reference network}}$$

Formula 1: Sensitivity of gene network

$$\text{Specificity} = \frac{\#\,\text{correctly estimated edges}}{\text{all estimated edges}}$$

Formula 2: Specificity of gene network

The greater the number of correct edges, the better the sensitivity. A higher value for sensitivity and specificity indicates better performance of the gene network.

3. Bayesian network algorithms

The gene networks of the yeast cell cycle were constructed using Bayesian network algorithms from data recorded in four different microarray experimental datasets. Five computer algorithms were developed in the construction of these gene networks. Among them are the Power Constructor (PC) algorithm, Hill Climbing (HC) algorithm, Maximum-Weight Spanning Tree (MWST) algorithm, K2 algorithm and MWST+K2 algorithm. Table 1 shows a comparison of these five algorithms.

K2 algorithm	K2 is the most widely used algorithm in Bayesian network structure learning. It is well known as a general method for inferring inter-node relations in a given node group based on a complete database free of missing data [22].
MWST algorithm	The MWST algorithm was developed by Chow and Liu [4]. This algorithm searches for an optimal tree structure by using the computed mutual information as edge weights [4]. The MWST associates a weight to each connection, where each weight represents the mutual information between the two variables. When the weight matrix is created, the MWST algorithm gives an optimal tree structure.
K2+MWST algorithm	Combining the K2 and MWST algorithms provides a better quality of network by speeding the execution efficiency. A known order of nodes is first calculated using the MWST algorithm, and these results are then used in K2.
Hill climbing algorithm	The sub-optimal hill climbing method is the heuristic K2 algorithm. The method focuses solely on precision and computation time at the expense of reliability, and it mainly relies on local exploitation. The more intensive the local exploitation, the stronger the need for specialized information about the function to be minimized [15, 16].
PC algorithm	PC is one of several dependent-based algorithms. This algorithm has an intuitive basis, and under some ideal conditions, it guarantees a graph that is equivalent to a true model of the data. It can be considered a smart selection and can intelligently order the questions needed to recover a causal structure.

Table 1. Comparison of five Bayesian network algorithms

Before constructing a gene network, it is necessary to preprocess the gene expression data. The gene expression data in Spellman's experiment is first normalized into the value of log2. We then categorize these values into three classes based on Friedman's threshold value of 0.5. These classes are represented by 3 discrete values: under-expressed (-1), normal expressed (0), and over-expressed (+1). In this section, we use the K2 algorithm to demonstrate how to construct a gene network. K2 is a search and score algorithm. Initially, a node order is set. Since the quality of the network structure is sensitive to the order of the nodes, an estimation of the nodes ordering is important. At first, the initial state of every node does not include the parent nodes. We use formula 3 to appraise whether the set of parent nodes, π_i, belongs to variable i. By finding every variable (node) that maximizes $g(i,\pi_i)$, we maximize the probability of Bayesian network structure B_s belonging to data D. The algorithm will stop when there are no parent nodes could increase the score.

$$g(i,\pi_i)=\prod_{j=1}^{q_i}\frac{(r_i-1)!}{(N_{ij}+r_i-1)!}\prod_{k=1}^{r_i}N_{ijk}!$$

Formula 3: Estimating function

As an example of how to calculate the partial conditional probability among genes, the data calculated between gene CLN2 and gene RNR3 in CDC15 conditions is computed and shown on Table 2. From Table 2, the conditional probability of P(RNR3|CLN2) can be expressed as P(-1|-1) = 0.0, P(0|-1) = 0.0, P(1|-1) = 0.556 , P(-1|0) = 0.5, P(0|0) = 0.444 , P(1|0) = 0.444 , P(-1|1) = 0.5 , P(0|1) = 0.556, P(1|1) = 0. The conditional probability is 0.0 when both RNR3 and CLN2 are under-expressed as well as 0.444 and 0.0 when both RNR3 and CLN2 are normal expressed and over-expressed. The Bayesian Gene Networks are then generated from these values using five algorithms.

	RNR3					**CDC5**			
CLN2		-1	0	1	**ACE2**		-1	0	1
	-1	0	0	0.556		-1	1	0	0
	0	0.5	0.444	0.444		0	0	0.909	0.375
	1	0.5	0.556	0		1	0	0.091	0.625
	CLB2					CLN2			
CLN1		-1	0	1	**CLB2**		-1	0	1
	-1	0	0.143	0.6		-1	0	0.25	0.625
	0	0.286	0.714	0.3		0	0.125	0.375	0.375
	1	0.714	0.143	0.1		1	0.875	0.375	0

Table 2. Partial conditional probability tables of genes in CDC15 dataset

4. GNGS system implementation

During this study, we developed a gene network generating system (GNGS) that is capable of generating the gene networks of the yeast cell cycle from experimental microarray data as well as comparing the performance of gene networks using five different Bayesian network algorithms. GNGS utilizes both MatLab's powerful processing ability and LabVIEW's dynamic interfaces in a single platform.

4.1 System architecture

GNGS is an innovative system that generates gene networks of the yeast cell cycle using Bayesian network algorithms. LabVIEW is the abbreviation for Laboratory Virtual Instrument Engineering Workbench. It is a kind of graphical programming language, or G language. The difference between the LabVIEW language and a general programming language is that LabVIEW can be written through encoded icons and can be used to construct a system. LabVIEW is an entirely graphical language which looks somewhat like an electronic schematic. It is hierarchical in that any virtual instrument that you design can be quickly converted into a module which can be a sub-unit of another virtual instrument (VI).

For example, every icon in Figure 1 has its own function. Programmers design the system by establishing connections between icons. Different data types can be expressed using color variations in the lines.

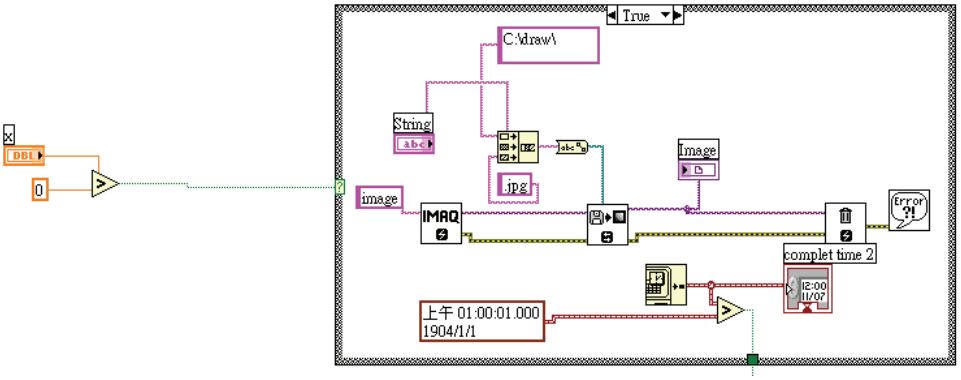


Figure 1. The VIs of displaying network function and executing time

MatLab, meanwhile, is an interactive, matrix-oriented programming language that enables us to express our mathematical ideas very concisely and directly; it considerably reduces development time and keeps code short, readable, and fully portable.

This system converts code written in MatLab and integrates the results onto the LabVIEW interface. In Figure 2 we show a Matlab scrip node (a VI provided by LabVIEW) which is the kernel that integrates the MatLab and LabVIEW languages together. The results computed from the Matlab scrip node can be seen as a VI output that transmits to another VI.

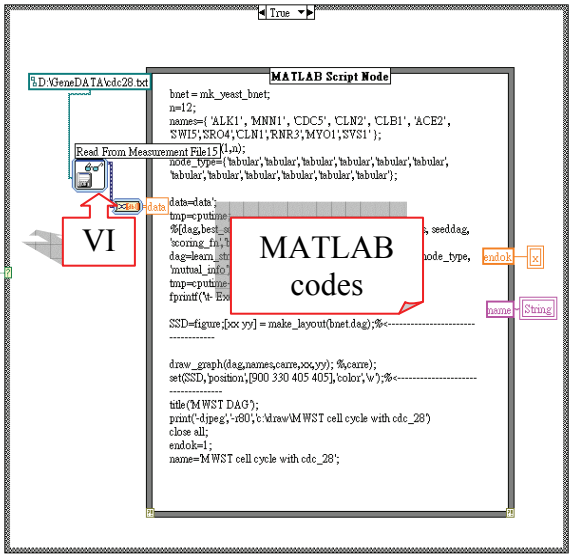


Figure 2. The Matlab scrip node generated from LabVIEW

4.2 System flow path

Figure 3 shows an overall flow path in the design of this system. A Select interface will be displayed to guide the users in running the program. By selecting the desired algorithm, GNGS will load the gene data and the system-constructed network from the selected gene dataset. After the gene network is displayed, the performance will then be calculated and shown in the summary table.

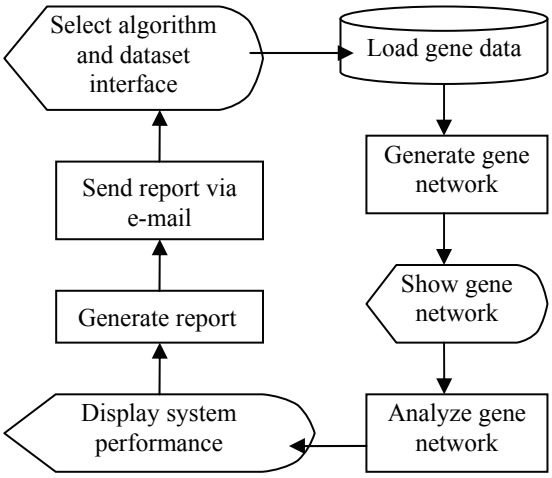


Figure 3. Design flow path of the system

5. System demonstration

The main functions of the GNGS interface include:

1. Algorithm selection section: users select the desired algorithm.
2. Network selection section: users select a desired network.
3. Dataset selection section: only displayed if the cell cycle button is clicked.
4. Execute icon: users click to run the program.
5. Clear icon: users click to clear all selections and release memory space.
6. Gene network display section: displays the resulting gene network.
7. Status slide bar section: shows the current execute status.
8. Summary table: displays summaries of users' requests.
9. Report icon: users click to generate reports that include the network graph and summary table.
10. Send icon: the system will send a report to the user through e-mail.
11. Exit icon: exits the interface.

When the users select one of the five Bayesian Network algorithms, the available network will be automatically displayed. Furthermore, if the users select cell cycle, four dataset types will then be displayed. After selecting the data set and clicking the Execute button, the light will turn to green as shown in Figure 4. The red light informs the user that the process is currently running. The slider bar on the right-hand side will show the execution status.

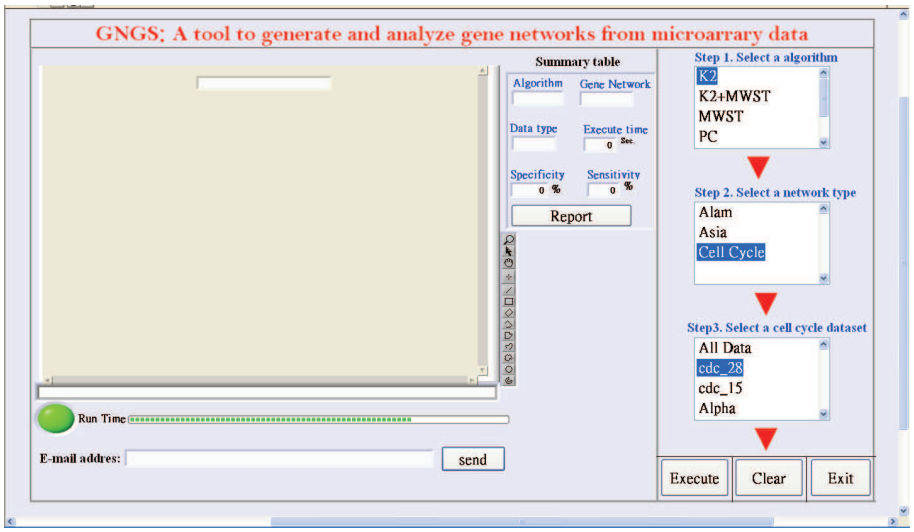


Figure 4. A screen of the system interface when the Execute icon is clicked

Figure 5 shows the final computation time and the gene network for the selected conditions. The result column will display information for the users, including algorithm, network type, dataset type, computation time, sensitivity, and specificity. The Clear button is also provided in case the users wish to clear the information. When the Clear button is clicked, the memory space will be released. Doing so assures that there are no memory overflow problems.

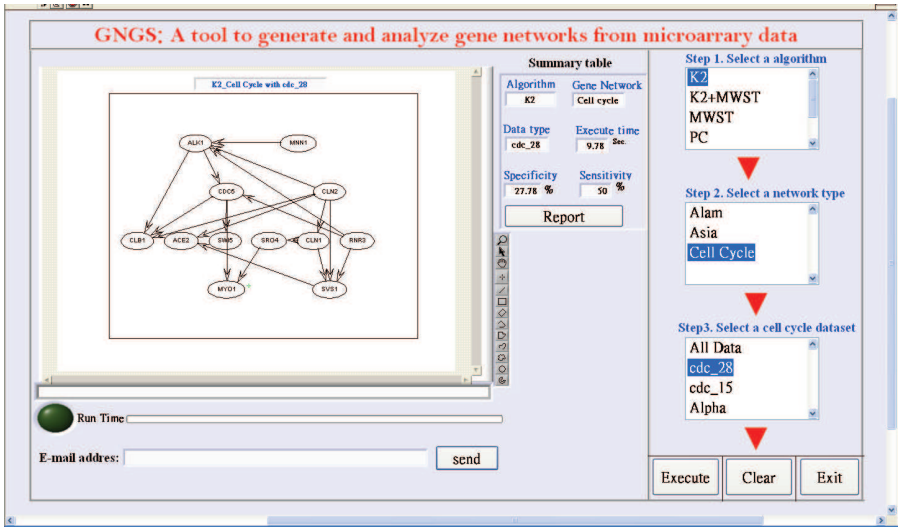


Figure 5. A screen of the system interface when the execution is finished.

6. Performance comparison

6.1 Alarm and Asia networks

Both the Alarm and Asia networks were used in this study to compare the performance for five different Bayesian network algorithms. Using the K2 algorithm as an example, the results of the Alarm and Asia network comparisons are shown in Figures 6 and Figure 7, respectively. In Figure 7, two structures are displayed: a known Asia network structure on the left and a K2-generated network on the right.

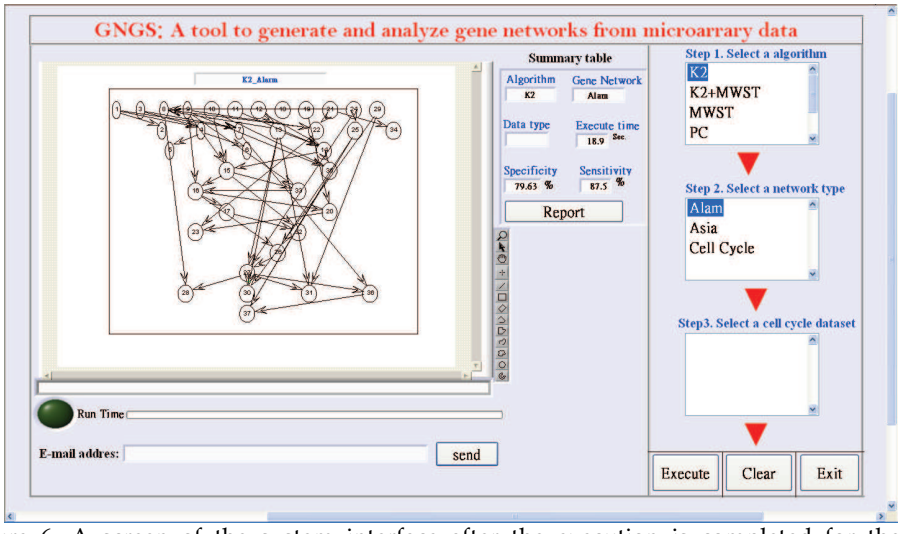


Figure 6. A screen of the system interface after the execution is completed for the K2 algorithm and the Alarm network.

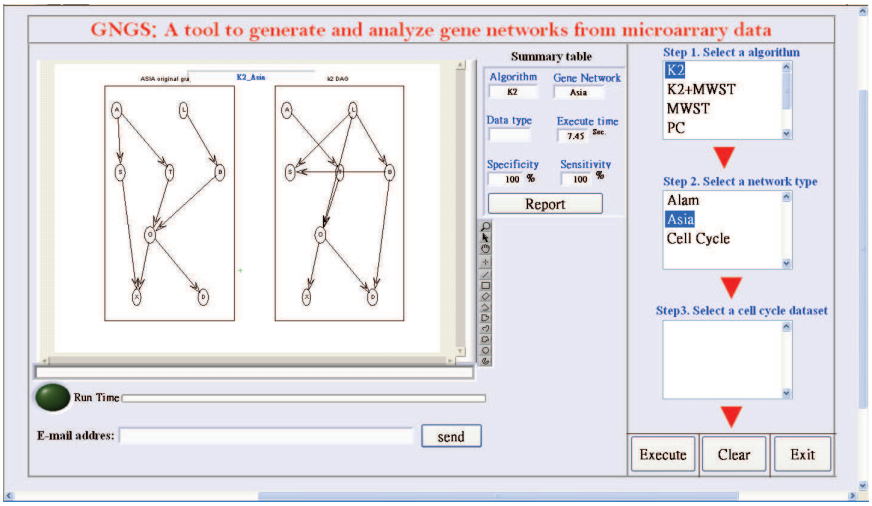


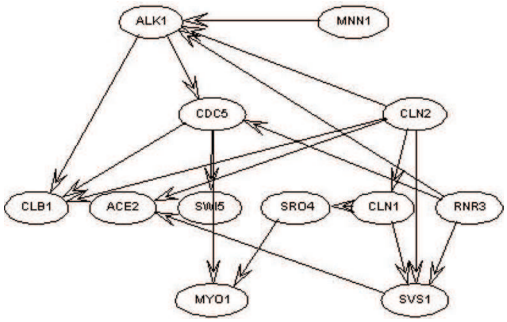
Figure 7. A screen of the system interface after the execution is completed for the K2 algorithm and the Asia network.

6.2 Gene networks of the yeast cell cycle

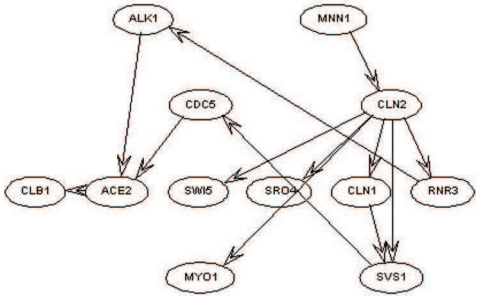
GNGS can generate gene networks based on the selected algorithm and dataset. In Figure 8, we show two gene networks generated from two experimental microarray datasets: one from the cdc_28 dataset and one from the cdc_15 dataset.

6.3 Comparison of computer execution time

In this section we compare the computer execution time for six datasets based on the K2 algorithm. The Alarm network, as expected, had the longest execution time at 28.1 seconds because it had the largest amount of data; gene networks from four cell cycle datasets all had an execution time of less than 1 second (Figure 9). The quantity of data within the dataset can affect the computer's execution time. Thus, we compared the system's execution time for five algorithms based on the same dataset. The times for K2, MWST, and K2+MWST were all less than one second. More complex search algorithms, such as the PC and HC algorithms, had a longer computer execution time; these, however, were still less than one minute (Figure 10).



(a)



(b)

Figure 8. Gene networks generated from the K2 algorithm by two different microarrary datasets: (a) the cdc_28 dataset, and (b) the cdc_15 dataset.

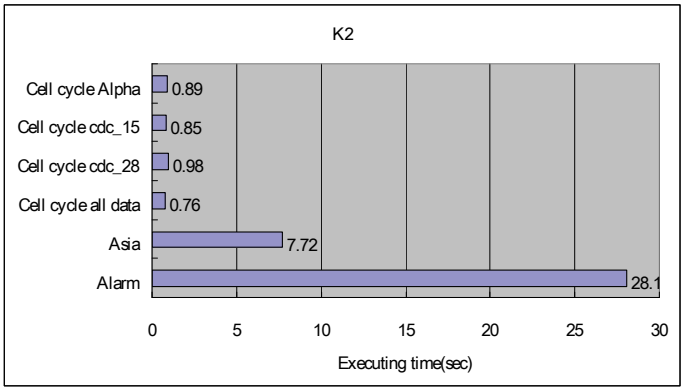


Figure 9. Comparison of execution time for 6 datasets based on the K2 algorithm

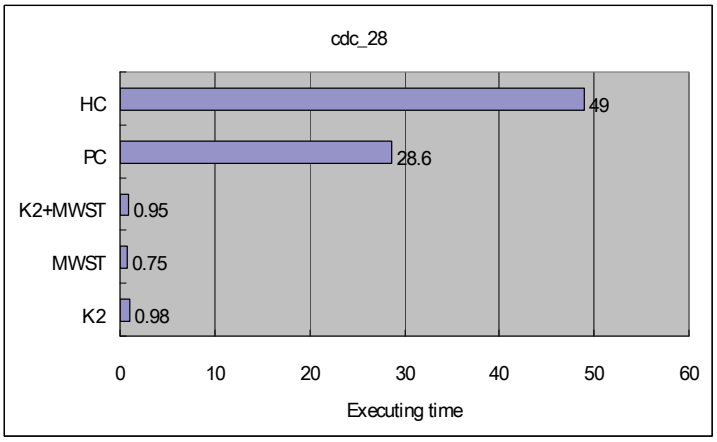


Figure 10. Comparison of execution time for 5 algorithms using cdc_28 dataset.

6.4 Comparison of sensitivity and specificity

By comparing the sensitivity and specificity of gene networks generated from six datasets using the K2 algorithm, it was found that the gene network constructed from the Asia dataset had the best performance (Figure 11). Both sensitivity and specificity were at 100 percent accuracy. The gene networks constructed from the cell cycle datasets, however, produced approximately 50 percent sensitivity.

In addition, we compared gene networks constructed based on five algorithms from the same dataset. It was notable that the HC algorithm required the longest computer time (Figure 10 and 12).

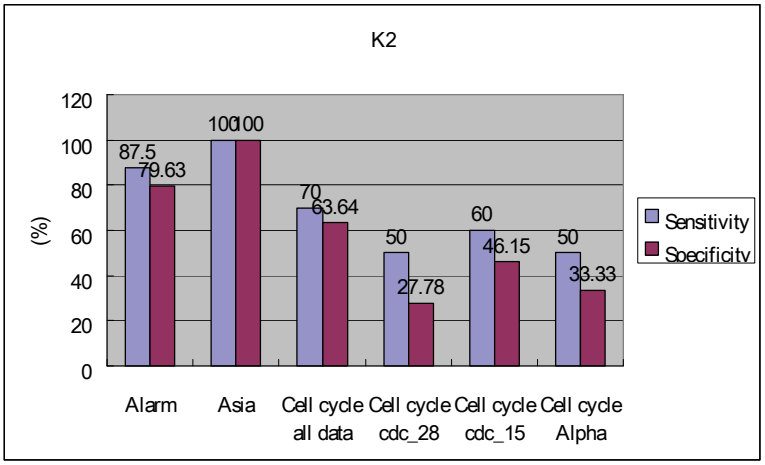


Figure 11. Comparison of sensitivity and specificity for 6 datasets based on the K2 algorithm

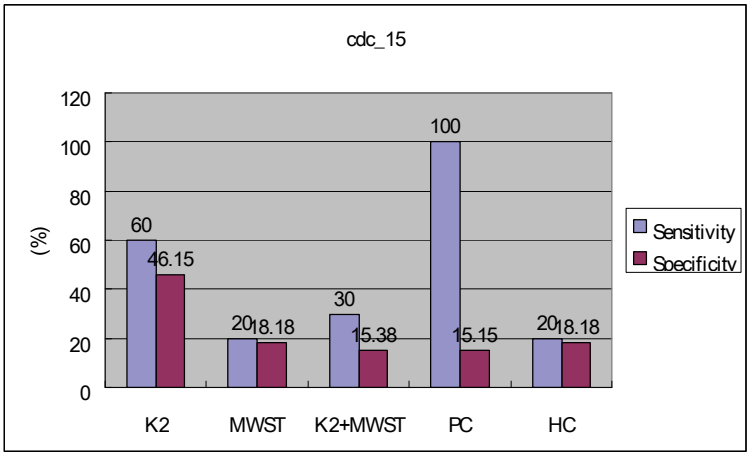


Figure 12. Comparison of sensitivity and specificity for 6 algorithms based on cdc_15 data set

6.5 Summaries of system performance

Finally, the system performance for all gene networks generated from the four experimental microarray datasets using five Bayesian network algorithms is summarized in Table 3. It was noted that the computer execution times for K2, MWST, and K2+MWST were all less than one second. Even for the more complicated operations such as the PC and HC algorithms, the execution time was still less than one minute.

A comparison of characteristics for each algorithm and its performance is summarized in Tables 3. Among these six algorithms, K2 has the best performance in terms of execution time, sensitivity, and specificity. In essence, the more nodes within a network structure (such as the Alarm network), the more run time is needed. This study found that the search and score method (K2) is the best strategy to find the optimal gene network due to its excellent performance and short execution time. The GNGS system is capable of running these algorithms, displaying the resulting networks, and analyzing system performance simultaneously.

Algorithm	Network	Data type	Executing time(sec)	Sensitivity	Specificity
K2	Alarm	-	28.1	87.5	79.63
	Asia	-	7.72	100	100
	Cell cycle	All data	0.76	70	63.64
		cdc_28	0.98	50	27.78
		cdc_15	0.85	60	46.15
		Alpha	0.89	50	33.33
MWST	Alarm	-	4.48	32.61	41.67
	Asia	-	7.92	62.5	71.43
	Cell cycle	All data	0.73	30	27.27
		cdc_28	0.75	20	18.18
		cdc_15	0.75	20	18.18
		Alpha	0.75	18.18	18.18
K2+MWST	Alarm	-	26.2	36.96	26.98
	Cell cycle	All data	0.9	30	27.27
		cdc_28	0.95	30	23.08
		cdc_15	0.9	30	15.38
		Alpha	1	30	15.38
PC	Asia	-	20.6	75	100
	Cell cycle	All data	20.7	100	15.15
		cdc_28	28.6	100	15.15
		cdc_15	28.9	100	15.15
		Alpha	28.5	100	15.15
HC	Asia	-	57	37.5	30
	Cell cycle	All data	55.1	20	18.18
		cdc_28	49	30	33.33
		cdc_15	51.4	20	18.18
		Alpha	52.7	30	27.27

Table 3. Summaries of system performance for all gene networks generated from four experimental microarray datasets using five Bayesian network algorithms.

7. Conclusion

In this paper, we have described a novel method to approach the study of gene networks. Firstly, we have developed and written five Bayesian network algorithms to construct gene networks of the yeast cell cycle based on four different microarray datasets. Secondly, we have implemented a gene network generating system that is more user-friendly. GNGS is capable of generating gene networks of the yeast cell cycle from experimental microarray data and comparing the performance of gene networks using five different Bayesian network algorithms. Our system utilizes both the powerful processing abilities of MatLab and the dynamic interface of LabVIEW in a single platform. Thirdly, we have compared the performance of each algorithm through measures such as execution time, sensitivity, and specificity for all five algorithms based on four different datasets.

In the near future, we intend to further improve performance by utilizing dynamic Bayesian network algorithms that more accurately reflect living cells' dynamic behavior. Our approach will then be used to explore the gene networks of human cells based on the microarray datasets of human cancers.

8. References

Beal et al (2005) A Bayesian approach to reconstructing genetic regulatory networks with hidden factors, *Bioinformatics, Vol 21, No. 3, 349 – 356*

Beinlich, I.A., Suermondt, H.J., Chavez, R.M., and Cooper, G.F. (1989) The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks, *Technical Report KSL-88-84*, Knowledge Systems Lab, Medical Computer Science, Stanford University.

Chen, T., He, H.L., Church, G.M. (1999) Modeling Gene Expression with Differential Equations, *Proc. of Pacific Symposium on Biocomputing*, pp. 29-40.

Cho,R.J. et al. (1998) A genome-wide transcriptional analysis of the mitotic cell cycle". Mol. *Cell*, 2, 65–73.

Chow, C., and Liu, C., (1968) Approximating discrete probability distributions with dependence trees, *IEEE Transactions on Information Theory*, 14(3), 462–467.

Cooper, G.F., Herskovits, E. (1992) A Bayesian method for the induction of probabilistic networks from data. *Mach. Learning J.* 9, 309–347.

Dejori, J. (2002) Analyzing Gene-Expression Data with Bayesian Networks, MS Thesis, *Elektro- und Biomedizinische Technik Technische Universit¨*at Graz. 2002.

D'haeseleer, P, Liang, S, and Somogyi, R. (1999) Tutorial: Gene Expression Data Analysis and Modeling, *Pacific Symposium on Biocomputing '99* (PSB'99).

Eisen,M.B., Spellman,P.T., Brown,P.O., and Bostein,D. (1998) Cluster analysis and display of genome-wide expression patterns., *Proc. Natl Acad. Sc*i., USA, 95, 14863–14868.

Friedman, N., Linial, M., Nachman, I., and Pe'er, D. (2000) Using Bayesian networks to Analyze Expression data, *Journal of Computational Biology*, 7, 601-620.

Gevaert,O. et al. (2006) Predicting the prognosis of breast cancer by integrating clinical and microarray data with Bayesian networks. *Bioinformatics*, 22, 184–190.

Heckerman, D., Geiger, D. and Chickering, D.(1995) Learning Bayesian networks: The combination of knowledge and statistical data, *Machine Learning*, 20(3), 197-243.

Husmeier,D., Dybowski,R. and Roberts,S., eds (2005) Probabilistic modelling in bioinformatics and medical informatics. *Springer-Verlag,* London, UK.

Kim, S., Imoto, S., and Miyano, S. (2004) Dynamic Bayesian network and nonparametric regression for nonlinear modeling of gene networks from time series gene expression data, *Biosystems*, 75, 57-65.

Lauritzen, S.L., and Spiegelhalter, D.J. (1988) Local computations with probabilties on graphical structures and their application to expert systems, *J. Royal Statistical society B*, 50:154-227.

Ovalle-Mart´ınez, F.J., Gonz´alez, J.S., and Stojmenovi´c, I., (2004) A parallel hill climbing algorithm for pushing dependent data in clients-providers-servers systems, *Mobile Network and Applications*, 9:257–264.

Renders, J.-M. and H. Bersini (1994) Hybridizing genetic algorithms with hill-climbing methods for global optimization: Two possible ways, Proceedings of the First IEEE International Conference on Evolutionary Computation, pp. 312-317. IEEE Press.

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

Spellman, P.T., et al. (1998) Comprehensive Identification of cell cycleregulated genes of the yeast saccharomyces cerevisiae by microarray hybridization, *Molecular Biology of the Cell*, 9, 3273-3297.

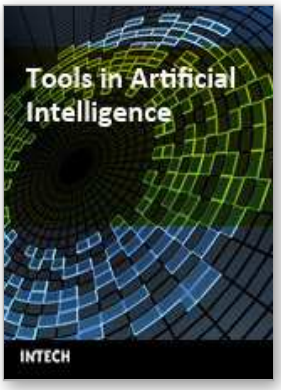
Tamayo, P. et al. (1999) Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation, *Proc. Natl Acad. Sci.*, USA, 96, 2907–2912.

Tavazoie, S. et al. .(1999) Systematic determination of genetic network architecture. *Nat. Genet*., 22, 281–285.

Thomas H. Cormen, Charles E. Leiserson, and Ronald R. Rivest (1990) Introduction to Algorithms., *MIT* Press.

Wright, R. and Yang, Z. (2004) Privacy-preserving Bayesian network structure computation on distributed heterogeneous data, In 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), Seattle, WA, USA.

Zhu, G. et al. (2000) Two yeast forkhead genes regulate the cell cycle and pseudohyphal growth." *Nature*,406, 90–94.



Tools in Artificial Intelligence

Edited by Paula Fritzsche

ISBN 978-953-7619-03-9
Hard cover, 488 pages
Publisher InTech
Published online 01, August, 2008
Published in print edition August, 2008

This book offers in 27 chapters a collection of all the technical aspects of specifying, developing, and evaluating the theoretical underpinnings and applied mechanisms of AI tools. Topics covered include neural networks, fuzzy controls, decision trees, rule-based systems, data mining, genetic algorithm and agent systems, among many others. The goal of this book is to show some potential applications and give a partial picture of the current state-of-the-art of AI. Also, it is useful to inspire some future research ideas by identifying potential research directions. It is dedicated to students, researchers and practitioners in this area or in related fields.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821