Network Dynamics of Budding Yeast Cell Cycle

CAI Chunhui

02050145

A thesis submitted in partial fulfillment of the requirement

for the degree of

Bachelor of Science (Honors)

Supervisor: Dr Tang Lei-han

Department of Physics

Hong Kong Baptist University

April 22nd, 2005

Declaration

I hereby declare that this thesis represents my own work which has been done in the past one year for the fulfillment for the degree of Bachelor of Science (Honors) in Physics major (computer science concentration) at Hong Kong Baptist University, and has not been previously included in a thesis, dissertation submitted to this or other institution for a degree, diploma or other qualification.

Signature: _____

Date: April 22, 2005

Acknowledgement

I would like to express my sincere gratitude towards my research project supervisor, Dr Tang Lei-han for sharing his precious time to supervise and patronize my project.

I also would like to thank Prof. S.Y Zhu as well as all the other team members for their valuable suggestion and comments.

I would thank all my friends, especially my girl friend for her continuous encouragement.

Finally, I would like to thank my family for their many years' caring and support.

ABSTRACT

The cell cycle of budding yeast is the succession of events that lead to reproduction of daughter cells and are tightly regulated. In this project, we study a much simplified model of cell cycle progression by Tang Chao and corroborate dynamic properties of the model, such as stability and robustness, with time course gene expression data. The regulation of yeast cell cycle clock is mainly based on the transcriptional regulation of cell cycle genes which are controlled by nine known transcription factors. Expression data are used to analyze the profiles of cell cycle genes. Fourier transform method is applied to sort genes according to their expression peak time. Furthermore, by combining expression time course data with Gerstein's regulatory network, we find three TF_orf clusters that function in different cell cycle stages.

Keywords: cell cycle, gene expression, regulation, transcription factor, regulatory network

Table of Content

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Yeast Biology

Budding Yeast, *Saccharomyces cerevisiae*, has been studied experimentally as a model organism of biology since the 1930's. Its complete genomic sequence was published in 1996, the first among eukaryotic organisms. The genome of *Saccharomyces cerevisiae* is divided up into 16 chromosomes, ranging from 220 kb to 2200 kb, with a total genome size of approximately 12,000kb. 6,183 open reading frames (ORFs) on the genome have been identified, most of which are believed to encode specific proteins.

Saccharomyces cerevisiae is a unicellular organism which, unlike more complex eukaryotes, can grow on defined media, giving the investigator complete control of environmental parameters. Moreover, since there are substantial cellular functions which are highly conserved from yeast to mammals, sequence information obtained in the yeast genome project is extremely useful as a reference against the sequences of human, animal or plant genes. The unique properties of the yeast *Saccharomyces cerevisiae*, among some 700 yeast species and its enormous hidden potential which has been exploited for many thousands of years, made it a preferred organism for research.

1.2 Introduction to Yeast Cell Cycle

A yeast cell receives a wide variety of cellular and environmental signals, which are often processed to generate specific genetic response. Here, we explore the molecular and genetic machinery of yeast cell cycle control which forms a highly independent system and is known in great detail.

The cell cycle is the succession of events whereby a cell grows and divides into two daughter cells that each contains the information and machinery necessary to repeat the process. The basic function of the yeast cell cycle is like other eukaryotic cells which is to duplicate accurately the vast amount of DNA in the chromosomes and then segregate the two copies precisely into mother cell and daughter cell. These processes define two major phases of the yeast cell cycle – S phase and M phase. Bud emergence and DNA duplication occur during S phase(S for synthesis). After S phase, DNA is segregated into mother cell and daughter cell (mitosis). When DNA has been partitioned, the cell undergoes cell division (cytokinesis), separating mother cell from the daughter cell. These two events occur in M phase. Besides these two major events, the cell requires much time to grow and double their mass of proteins and organelles. After S phase, replicated DNA is checked for its genetic integrity to ensure there is no damage during replication, otherwise the cell cycle is halted for DNA reparation. Hence, extra gap phases are inserted into the cell cycle – G2 between S phase and M phase; while G1 between M phase and next S phase. Thus, the cell cycle is divided into four sequential phases: G1, S, G2, M.

Basic biology on yeast and its cell cycle are covered in most textbooks: *Biochemistry* (Mathews et al 2000), *Molecular Biology of the cell cycle* (Alberts et al 2002) and *Gene VIII* (Lewin 2003). The online databases contain publications and comprehensive information on yeast: SGD (http://www.yeastgenome.org/), SMD

(http://genome-www5.stanford.edu/), CYGD (http://mips.gsf.de/proj/yeast/CYGD/db/) and KEGG (http://www.genome.jp/kegg/).

1.3 Objectives

Yeast cell cycle is tightly regulated for proper functioning at proper time. The cell cycle control system guarantees the stability and robustness when cell goes through cell cycle progression. In the present project, Prof. Tang Chao's dynamic model is employed and further implemented to demonstrate the global dynamic properties and stabilities of cell cycle network. Our aim is to understand how the command system, which is basically a controlled program of gene expression, is designed to regulate the cell cycle. Beyond this much simplified model that focuses on the cell-cycle progression, we wish to dig deeper into the genetic construction of regulatory circuit and the ensuing dynamics. For this purpose, we examine the expression pattern of the 792 known cell cycle genes from time course measurement. By clustering into phase-synchronized groups, we hope to decipher details of their regulatory program, including on-off switch by single or multiple TFs, and the associated dynamic process of TF binding or activation.

Chapter 2

Yeast Cell Cycle Control

2.1 Yeast Cell Cycle is Tightly Regulated

For many years, cell biologist watched the puppet show of DNA synthesis, mitosis and cytokinesis but had no idea of what lay behind the curtain controlling these events. The cell cycle control system was simply a black box inside the cell.

The cell cycle control system possesses the following features.

i. A clock, or timer, that turns on each event at a specific time, and provide a relatively fixed amount of time for the completion of each event;
ii. A mechanism for initiating events in correct order; for example, entry into mitosis must always come after DNA replication
iii. A mechanism to ensure that each event is triggered only once per cycle; for example, the DNA cannot be replicated twice during a single cell cycle
iv. Binary (on/off) switch that trigger events in a complete, irreversible fashion; It would be a disaster, if events like nuclear envelope breakdown were initiated but not completed.
v. Robustness, backup mechanism to ensure that the cycle can process smoothly even when parts of the systems malfunction
vi. Adaptability. So that the system's behavior can be modified to suit specific environmental conditions.

What makes up the control system to regulate the cell cycle clock? A lot of surveys and studies have been done by biologists, showing that the control system is mainly based on a family of protein kinase knows as CDK (cyclin-dependent kinase), and gene regulation. In budding yeast cell cycle, there is only one CDK – Cdc28 (C Wittenberg, 2005; Mart Loog and David O. Morgan, 2005; Alberghina et al, 2004).

Much is known about Cdc28 activities and its functions (). Cdc28 associates successively with different cyclins which is also proteins (Cln1,2,3,4; Clb1,2,5,6) to trigger the different events of the cycle, and its activity is usually terminated by cyclin degradation or inhibitory phophorylation (Nash et al., 1988). The activity of the Cdc28 rises and falls as cell progress through the cycle and the oscillations lead directly to cyclical change activation of certain proteins that initiate the major events of the cell cycle, for example an increase in Clb2/Cdc28 activity at the beginning of mitosis leads to increased activation of proteins that control chromosome condensation, nuclear envelope breakdown and spindle assembly. Others like Cln2/Cdc28 is responsible for DNA replication and Clb5/Cdc28 is responsible for bud emergence (Tyers et al., 1993; Schwob and Nasmyth, 1993).

The cell cycle control obviously depends on protein-protein interactions, which is also referred to as post-transcriptional mechanism. However, transcriptional regulation provides another level of control which is more fundamental. The genes peak in different phase during cell cycle are responsible for synthesis of cell cycle specific proteins. Some cyclin levels, for example, are controlled through cyclin gene transcription, since the genes mainly code for proteins.

2.2 Yeast Cell Cycle Dynamic Model

How do physicists study cell cycle regulatory process? As they first studied the hydrogen atom before coming into the more complex atoms, physicists first focused on the yeast cell cycle with the most simplified network. Prof. Tang Chao developed a simple dynamic model with just a few nodes to investigate the global dynamic properties and stabilities of cell cycle network (Li 2004). Fig 2.1 shows the proteins and their interactions (in the sense of information flow) in Chao's network.

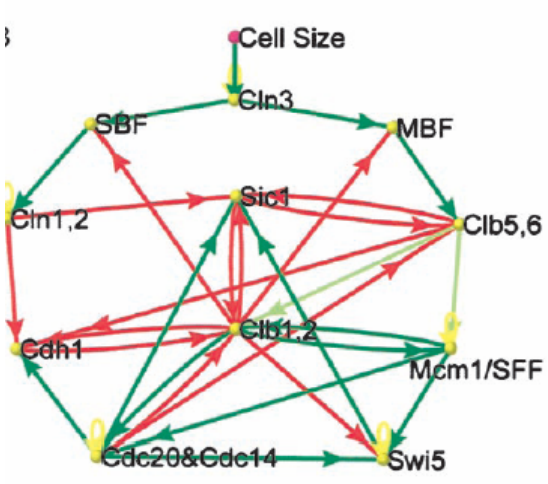

Fig 2.1: Chao's network

This regulatory network includes cell size check point and 3 classes of proteins: cyclin, which bind to the kinase Cdc28; the inhibitors, degraders, and competitors of the cyclin/Cdc28 complexes (Sic1, Cdh1, Cdc20, Cdc14) and transcription factors (SBF, MBF, Mcm1/SFF, Swi5). Green arrows represent positive regulations. For example, when the cell grows large enough, the Cln3/Cdc28 will be activated, which in turn activates a pair of transcription factor groups, SBF and MBF to activate the genes of the cyclins Cln1,2 and Clb5,6, respectively. Red arrows represent negative regulaton (inhibition, repression, or degradation). For example, the protein Sic1 can bind to the Clb/Cdc28 complex to inhibit its function, Clb1,2 phosphorylate Swi5 to prevent its entry into the nucleus, whereas Cdh1 targets Clb1,2 for degradation.

Yellow loops are add to represent 'self-degradation' to those nodes that are not negatively regulated by others. The degradation is modeled as a time-delayed interaction: if a protein with a self yellow arrow is active at time t (Si(t) =1) and if its total input is zero from time t to t+1, it will be degraded at time t+1, i.e., (Si=0). Since much of the biology seems to be reflected in the on–off characteristics of the network components, the nodes and arrows can be treated as logic-like operations in this simplified dynamic network. Hence, each node i has only two states, Si=1 and Si=0, representing the active and the inactive state of the protein, respectively, with totally 11 nodes in the network. The protein states in the next time step are determined by the protein states in the present time step via the following rule:

$$S_i(t+1) = \begin{cases} 1, & \sum_j a_{ij} S_j(t) > 0 \\ 0, & \sum_j a_{ij} S_j(t) < 0 \\ S_i(t), & \sum_j a_{ij} S_j(t) = 0 \end{cases}$$

where a_{ij}=1 for a green arrow from protein j to protein i and a_{ij} =1 for a red arrow from j to i.

Fixed Points

We implemented the dynamic model to study the time evolution of the protein states. Following Tang Chao's work, we start from each of the 2,048 initial states in the 11-node network. We find that all of the initial states eventually flow into one of the seven stationary states (fixed points) shown in Table 2.1. There is one big fixed point attracting 1,764 states among the seven fixed points. This super stable state is the biological G1 stationary state. The advantage for a cell's stationary state to be a big attractor of the network is obvious: the stability of the cell state is guaranteed since under normal conditions the cell will be sitting at this fixed point, waiting for the signal for another round of division.

Basin size	Cln3	MBF	SBF	Cln1,2	Cdh1	Swi5	Cdc20	Clb5,6	Sic1	Clb1,2	Mcm1
1,764	0	0	0	0	1	0	0	0	1	0	0
151	0	0	1	1	0	0	0	0	0	0	0
109	0	1	0	0	1	0	0	0	1	0	0
9	0	0	0	0	0	0	0	0	1	0	0
7	0	1	0	0	0	0	0	0	1	0	0
7	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0

Table 2.1: Fixed points of cell-cycle network

Biological Pathway

Next, we start the cell-cycle process with the cell size signal, and observe that the system starts from G1 and goes back to the G1 stationary state. The temporal evolution of the protein states indeed follows the cell-cycle sequence, shown in Table 2.2. This is the biological trajectory or pathway of the cell-cycle network which represents the cell cycle progression

.

Time	Cln3	MBF	SBF	Cln1,2	Cdh1	Swi5	Cdc20	Clb5,6	Sic1	Clb1,2	Mcm1	Phase
1	1	0	0	0	1	0	0	0	1	0	0	Start
2	0	1	1	0	1	0	0	0	1	0	0	G1
3	0	1	1	1	1	0	0	0	1	0	0	G1
4	0	1	1	1	0	0	0	0	0	0	0	G1
5	0	1	1	1	0	0	0	1	0	0	0	S
6	0	1	1	1	0	0	0	1	0	1	1	G2
7	0	0	0	1	0	0	1	1	0	1	1	M
8	0	0	0	0	0	1	1	0	0	1	1	M
9	0	0	0	0	0	1	1	0	1	1	1	M
10	0	0	0	0	0	1	1	0	1	0	1	M
11	0	0	0	0	1	1	1	0	1	0	0	M
12	0	0	0	0	1	1	0	0	1	0	0	G1
13	0	0	0	0	1	0	0	0	1	0	0	Stationary G1

Table 2.2: Time evolution of cell-cycle pathway

Newborn daughter cells grow to a critical size to have enough Cln3 to activate the transcription factors, MBF and SBF, which transcriptional activate two classes of cyclins, Cln1,2 and Clb5,6. Cln2 is primarily responsible for bud emergence and Clb5 for initiating DNA synthesis. Clb5-dependent kinase activity is not immediately evident because the G1-phase cell is full of cyclin-dependent kinase inhibitors (Sic1). After the Sic1 is phosphorylated by Cln2/Cdc28 for degradation, Clb5/Cdc28 is released to do its job.

Another class of cyclin, Clb2, are out of the picture in G1 because their transcription factor, Mcm1, is inactive, their degradation pathway, Cdh1/APC, is active, and their stoichiometric inhibitors, Sic1, are abundant. Cln2- and Clb5-dependent kinases remove Sic1 and inactivate Cdh1. Clb2 is then allowed to appear. Moreover, Clb2/Cdc28 soon activates its own transcription factor, Mcm1, which in turn further drive its synthesis.

Clb2/Cdc28 turns off SBF and MBF. As Clb2/Cdc28 drives the cell into mitosis, it also sets the stage for exit from mitosis by stimulating the synthesis of

Cdc20 which is transcriptional regulated by Mcm1. Then, Cdc20 promotes the activation of Cdc14. Cdc20&Cdc14 play several roles in mitotic exit. First, they degrades Clb5,6 and Clb1,2, remove their potency on Cdh1 inactivation. Next, they activate Cdh1, stabilizing Sic1, and activate Swi5 (the transcription factor for Sic1). As Clb2-kinase activity is quenched by Cdh1 and by Sic1 to below a threshold value, a signal for exit from mitosis is triggered, the cell divides and returns to G1 phase.

To investigate the dynamical stability of cell cycle pathway, Li et al (Li, 2004) analyzed the dynamic trajectories of all 1,764 protein states that will flow to the G1 fixed point, shown in Fig 2.2. The cell cycle pathway is colored in blue and so is the node representing the G1 stationary state. The dynamic flow of the protein states is convergent onto the biological pathway, making the pathway an attracting trajectory of the dynamics. With such a topological structure, the cell-cycle pathway is a very stable trajectory; it is very unlikely for a sequence of events, starting at the beginning (or at any other point) of the cell-cycle process, to deviate from the cell-cycle pathway.

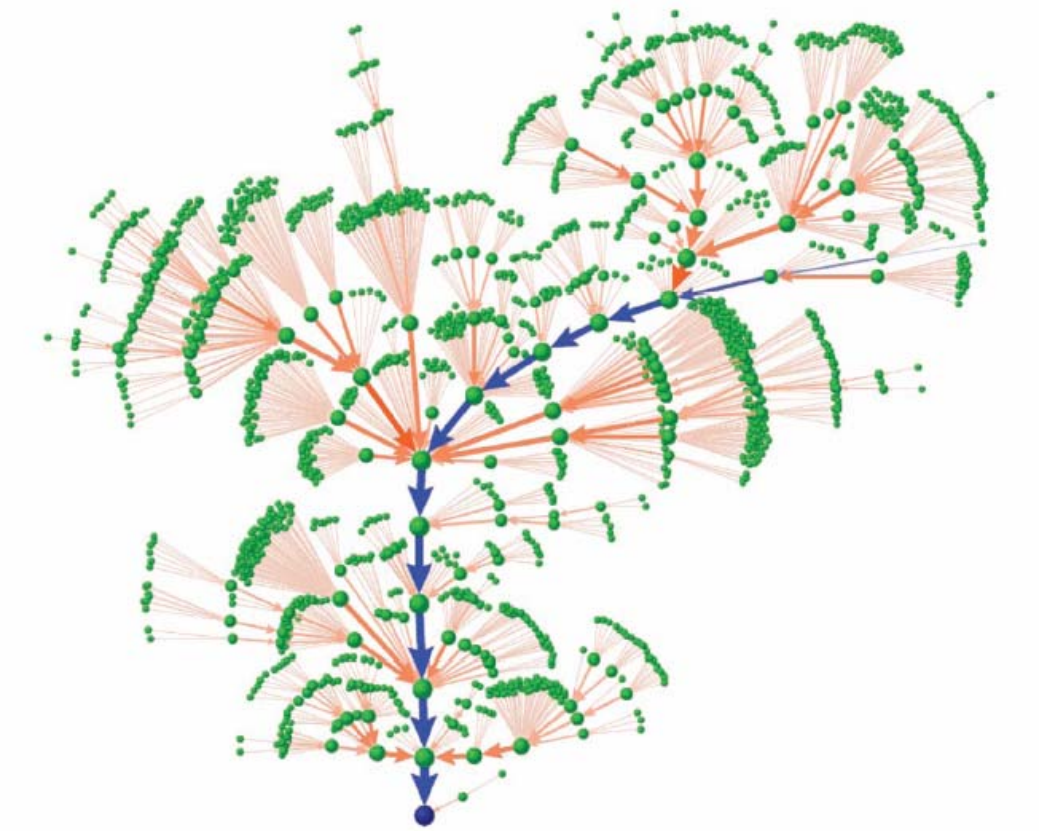

Fig 2.2: Dynamic trajectory of 1,764 protein states

From Li et al (Li 2004), the yeast cell-cycle network is robustly designed. Furthermore, since the network is only a skeleton of a larger cell-cycle network with many ‘‘redundant’’ components and interactions, the complete network is expected to be even more stable.

Chapter 3

Gene Expression and Transcriptional Regulatory Network

3.1 Gene Expression

Regulation of the cell cycle clock is mainly effected through a controlled program of gene expression (Paul T. Spellman, et al, 1998). In budding yeast, there are about 800 cell cycle genes, oscillating during the cell cycle. Some of these genes encode proteins with known cell-cycle functions, such as cell cycle control, cell wall biogenesis, DNA replication and so on, but most are unknown.

One set of gene expression time course data of cell cycle category was collected from Stanford Microarray Database (http://genome-www5.stanford.edu). The expression ratio was measured under the specific condition that yeast cells were blocked in mitosis using a cdc15-2 temperature sensitive mutant at restrictive temperature which is in order to synchronize the sample cells. The mutant can prevent the release of CDC14 which will further activate CDH1. Hence, the cells are arrested in 10^{th} time step of Tang Chao's cell cycle pathway. The culture was then shifted to permissive temperature (25^{o}C), and released into the cell cycle. The cell cycle then starts at M/G1. Samples were then taken every 10 mins (some are taken every 20 mins) during the course of over two full cell cycles, 290 mins (Table3.1). Then, we collected 798 cell cycle genes with some of predicted phases according to Richard Young's work in MIT (http://web.wi.mit.edu/young/cellcycle/). After the combination of Richard Young's cell cycle genes with those involved in the time course data, we finally have a data set of 792 cell cycle genes.

Time point	Sample taken time
1	cdc15 010 min
2	cdc15 030 min
3	cdc15 050 min
4	cdc15 070 min
5	cdc15 080 min
6	cdc15 090 min
7	cdc15 100 min
8	cdc15 110 min
9	cdc15 120 min
10	cdc15 130 min
11	cdc15 140 min
12	cdc15 150 min
13	cdc15 160 min
14	cdc15 170 min
15	cdc15 180 min
16	cdc15 190 min
17	cdc15 200 min
18	cdc15 210 min
19	cdc15 220 min
20	cdc15 230 min
21	cdc15 240 min
22	cdc15 250 min
23	cdc15 270 min
24	cdc15 290 min

Table 3.1 Time elapsed after escape from cdc15 arrest

In order to standardize and analyze the data set, we normalized the gene expression data so that the average log2(ratio) over the course of the experiment is equal to 0 and were further divided by the standard deviation. Fig 3.1 shows the raw data set using color coding of the gene expression value during the yeast cell cycle. Genes correspond to rows, and the time points of the experiment are the columns. Yellow color means positive regulation, while blue color means negative regulation.

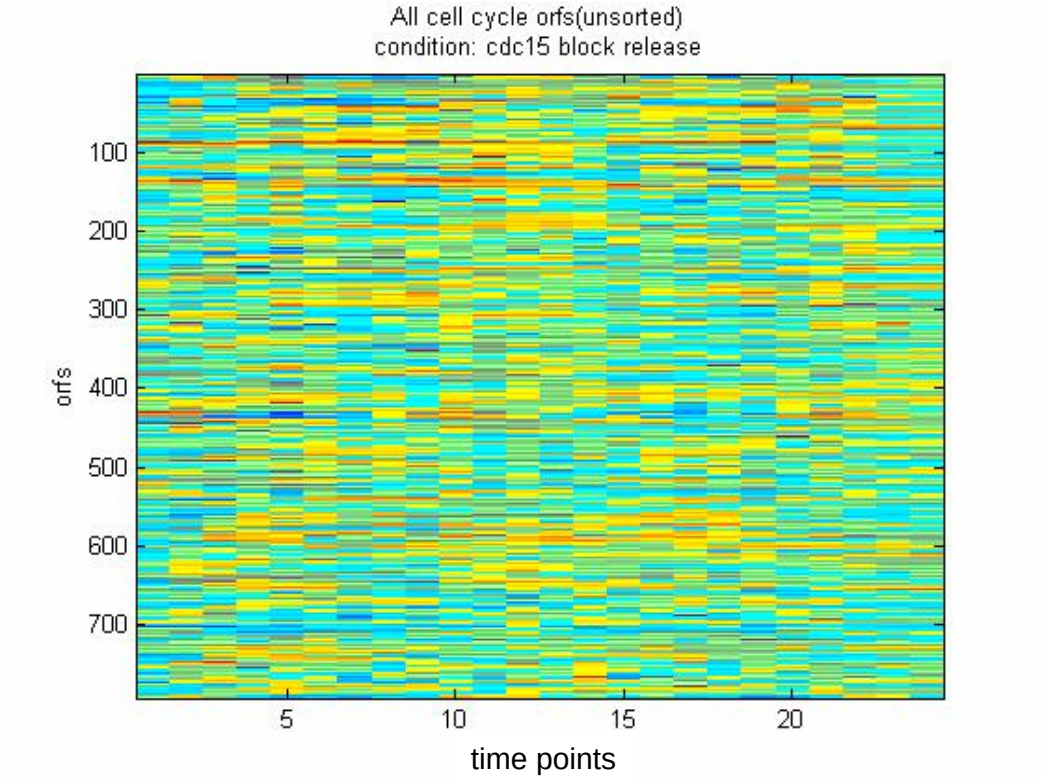

Fig 3.1: Raw data of all cell cycle genes

Hardly any information can be extracted from the above figure, because the genes are disorderly placed along the Y axis. In order to test if these data is good and whether we can see some periodic patterns that are expected, we applied Fourier Transform to analyze the data. We calculated Fourier Transform magnitudes for each gene in terms of different values of omega.

$$Im = \sum \sin(\omega t_j) x(t_j) \qquad (1)$$

$$Re = \sum \cos(\omega t_j) x(t_j) \qquad (2)$$

$$I = A^2 + B^2 \qquad (3)$$

$$\Phi = \tan^{-1}(Im/Re) \qquad (4)$$

Since the cell cycle period usually varies from 90mins to120mins. The test omega value is taken as 2*pi/90. Fig 3.2 shows the distribution of Fourier transform magnitudes, with omega ranging from 0.4*test to 1.5*test and we can easily see that

most genes peak at either position 4 or position 5, which is 115 min and 125 min respectively. Thus the periodical property of these genes under this condition is quite similar. We also found that Fourier Transform magnitudes are unstable for small variations of omega, so the magnitudes were averaged as well as phase over 115 to 125 mins with 1 min each step to get both magnitudes and phase of each gene.

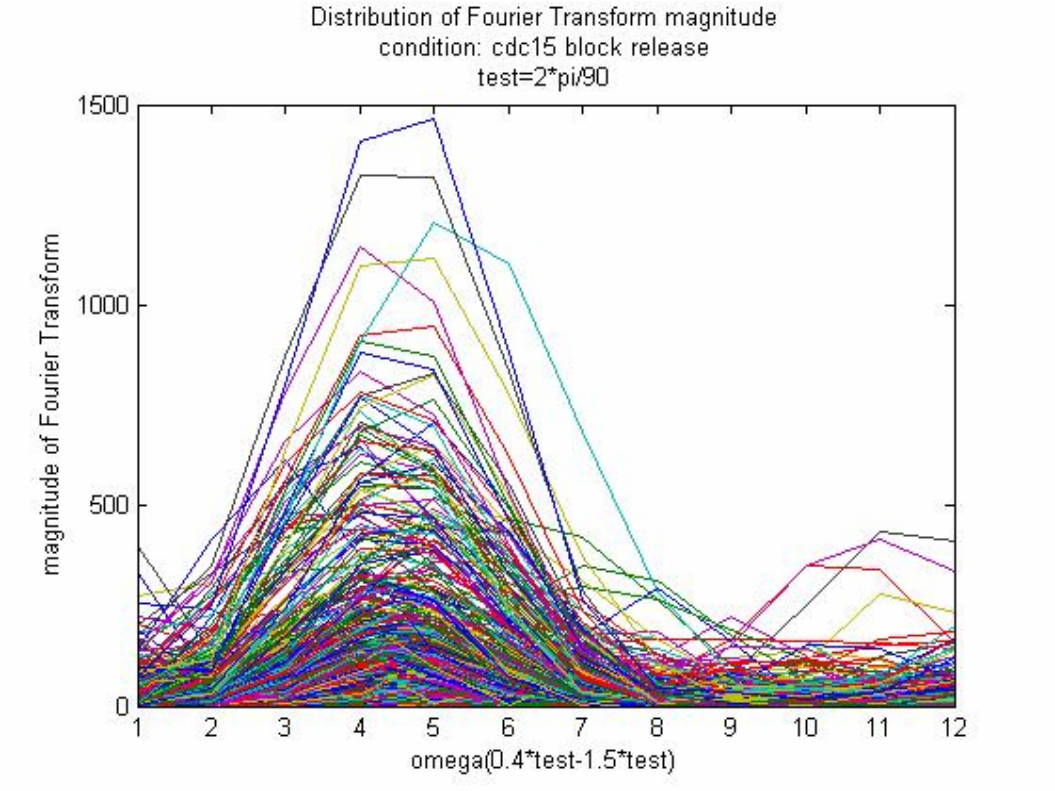

Fig 3.2: Fourier transform magnitudes distribution

The genes are then sorted according to their magnitudes which are their fluctuation strength, and further sorted in terms of their phase (time of peak expression). Fig 3.3 shows results of sorting the strongest 50, 100, 200, 400, 600 and all cell cycle genes respectively.

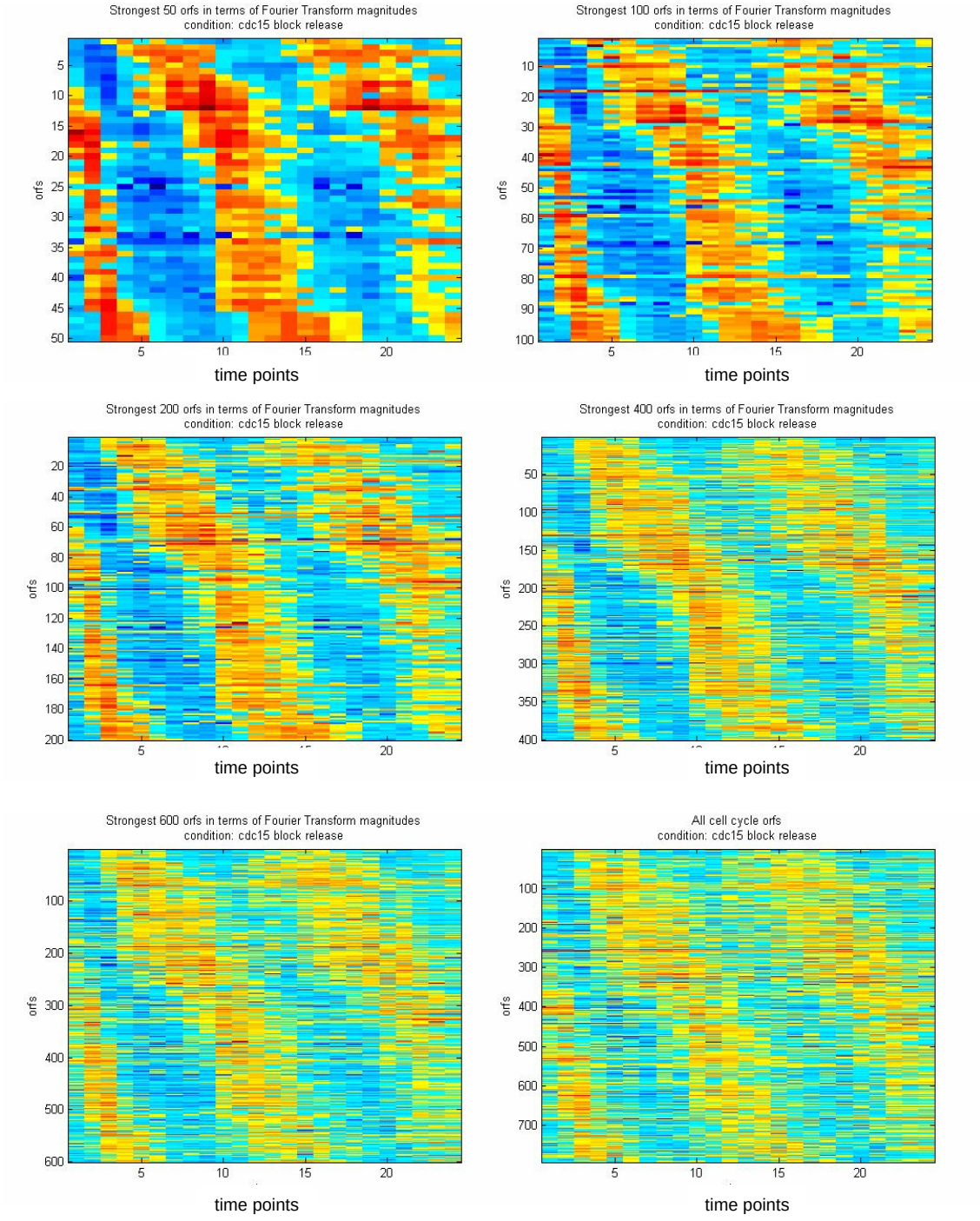

Fig 3.3: Expression of sorted genes (50,100,200,400,600,792)

Fig 3.3 shows clearly the periodic patterns of cell cycle genes. These genes are regulated in a periodic manner coincident with the cell cycle. Such gene regulation is required for proper functioning of the control mechanism to maintain events' order through cell cycle. For example, the occurrence of 50 genes of strongest oscillation obeys the cell cycle sequence, though the boundary is not absolute, shown in Table 3.1 and Fig 3.4.

Orf	Phase	Orf	Phase	Orf	Phase
YPL187W	m/g1	YJL078C	g1	YBR009C	s
YJL159W	m/g1	YJL115W	g1	YDR224C	s
YKL185W	m/g1	YPL267W	g1	YBR010W	s
YKL164C	m/g1	YBL035C	g1	YBL003C	s
YKL163W	m/g1	YLR286C	g1	YMR003W	s/g2
YDR261C	s	YDR097C	g1	YML052W	g2/m
YOR307C	g1	YGR189C	g1	YHL028W	g2/m
YLR079W	m/g1	YIL066C	g1	YPR149W	g2/m
YKR077W	g1	YBR088C	g1	YMR032W	g2/m
YBR158W	m/g1	YDL003W	g1	YLR190W	g2/m
YOR308C	g1	YHR143W	g1	YBR038W	g2/m
YNL327W	m/g1	YAR007C	g1	YBR092C	g2/m
YBR108W	g1	YBR089W	g1	YDR033W	g2/m
YLR049C	g1	YOL090W	g1	YDR225W	s
YGL028C	g1	YPL256C	g1	YBR054W	g2/m
YGR044C	g1	YOL007C	g1	YNL160W	m/g1
YCL024W	g1	YNL030W	s		

Table 3.2: 50 strongest oscillating genes

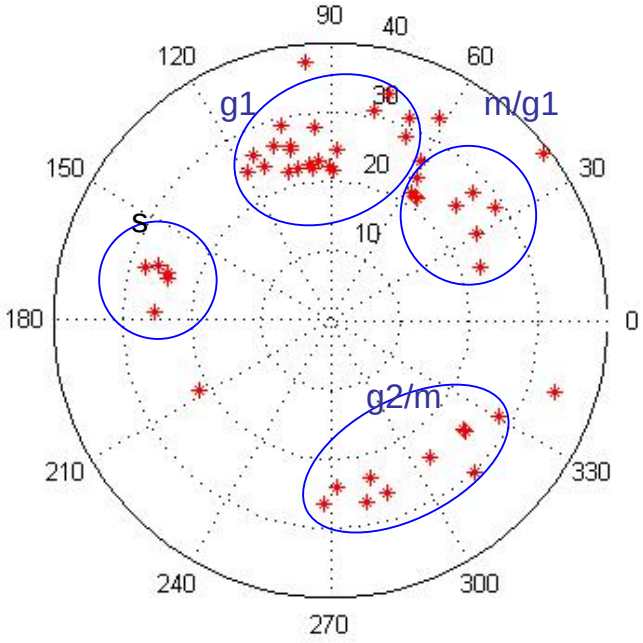

Fig 3.4: Polar distribution of 50 strongest oscillating genes

Besides the differences among their peak expression time, genes may have distinct duration of activation. Some genes are turned on for a short period and then turned off again, while others are turned on for a longer period. This may be related to their functionality. For example, genes responsible for cell wall biosynthesis and integrity should be active during the course of mitosis (YHL028C) while others required for cell cycle check points only need to function over a short period of time (YCL024W), shown in Fig 3.5.

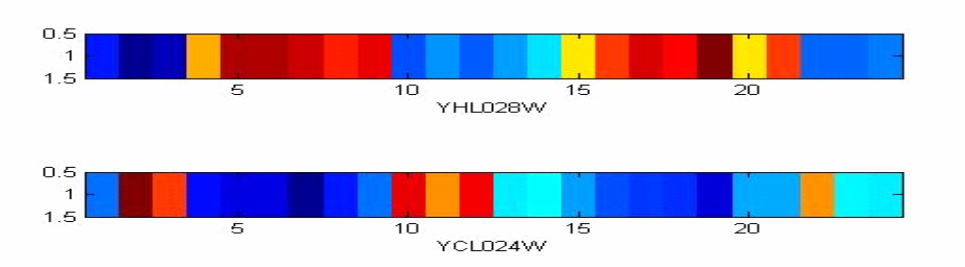

Fig 3.5: Genes' activation durations are related to their functions

The experiment as mentioned previously starts at late M phase and then the cell enters G1 very quickly. The literature review of yeast cell cycle elucidates G1 and M phases both generally take up about 1/3 cell cycle time, while S phase takes up over 1/2 of the remaining time (Dien BS, Srienc F, 1991). Since time duration of the cell cycle in this experiment is around 120 mins, we can assign each time point with a hypothetic phase description, i.e. 0 mins to 10 mins – M, 10 mins to 50 mins – G1, 50 mins to 80 mins – S, 80 mins to 90 mins – G2 and 90 mins to 120 mins – M, as shown in Table 3.3. Gene expression is further plotted along with the hypothetical phase description of first round of cell cycle in Fig 3.6. The result is supported by the 50 genes that have the strongest oscillations, as shown in Fig 3.4. The time for peak expression of m/g1, g1, s and g2/m genes is 5 mins to 20 mins, 20 mins to 40 mins, 50 mins to 60 mins and 90 mins to 110 mins reapectively.

Time point	Sample taken time	Hypothetical Phase
1	cdc15 010 min	M
2	cdc15 030 min	G1
3	cdc15 050 min	G1
4	cdc15 070 min	S
5	cdc15 080 min	S
6	cdc15 090 min	G2
7	cdc15 100 min	M
8	cdc15 110 min	M
9	cdc15 120 min	M
10	cdc15 130 min	M
11	cdc15 140 min	G1
12	cdc15 150 min	G1
13	cdc15 160 min	G1
14	cdc15 170 min	G1
15	cdc15 180 min	S
16	cdc15 190 min	S
17	cdc15 200 min	S
18	cdc15 210 min	G2
19	cdc15 220 min	M
20	cdc15 230 min	M
21	cdc15 240 min	M
22	cdc15 250 min	M
23	cdc15 270 min	G1

24	cdc15 290 min	G1

Table 3.3 Cell cycle progression time with phase description

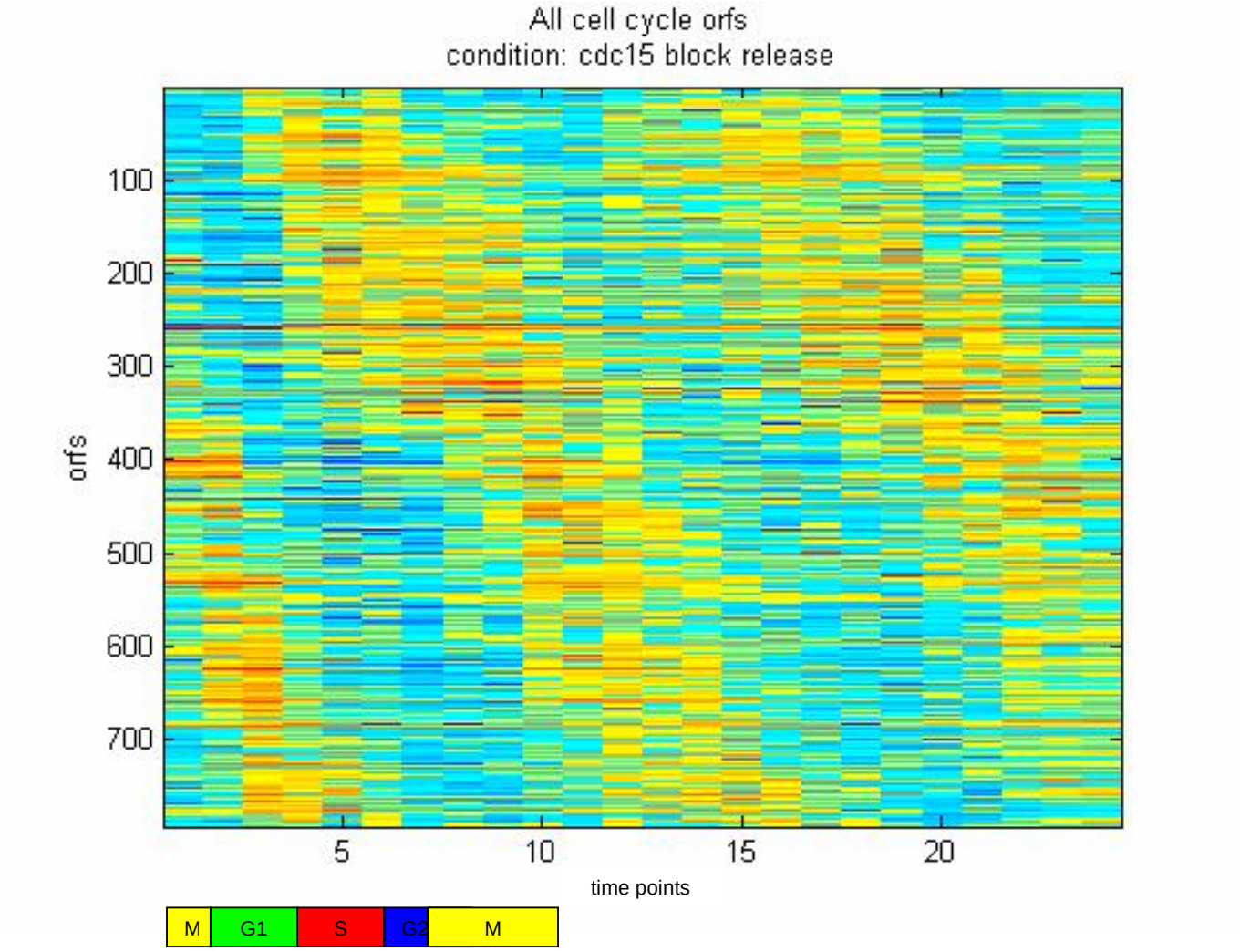

Fig 3.6: Hypothetic phase description

First round of cell cycle progression

2.2 Transcriptional Regulatory Network

Gene activation and repression events are initiated by a special class of proteins known as transcription factors (TF) which bind to short DNA sequence (known as binding motifs or TF binding sites) upstream the transcription start site of a given gene. The collection of TF's and their target ORF's forms a transcriptional regulatory network that constructs the backbone of the genetic regulatory program for all cellular process. The yeast cell cycle gene expression program can be viewed as outcome of such a program in action. Nine known cell cycle transcriptional factors are involved, each regulating a group of genes and functioning during one stage of cell cycle. Furthermore, these nine transcription factors are divided into three groups, as illustrated in Fig 3.7. Mbp1, Swi4 and Swi6 control the late G1 genes. Mcm1, Fkh2 (or Fkh1) and Ndd1 control the transcription of G2/M genes. Mcm1 is also involved in the transcription of genes at the end of m and early G1 together with Swi5 and Ace2 (Itamar Simon, et al, 2001).

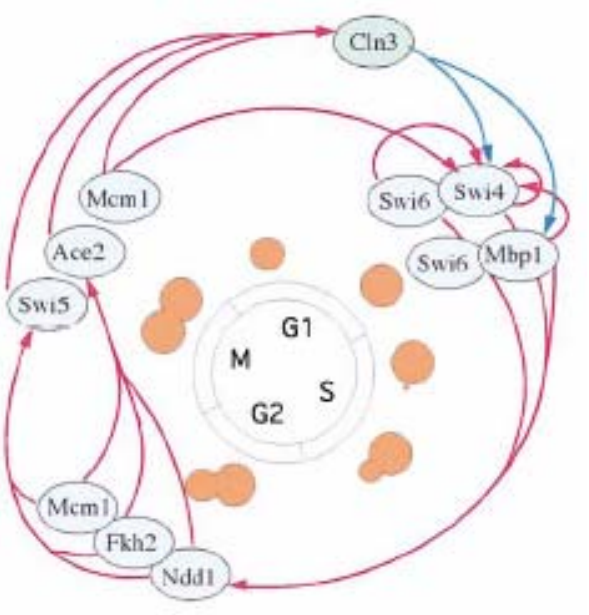

Fig 3.7: Groups of transcription factors function in different cell cycle stages

In 2004, a genome-scale transcriptional regulatory network, involving 7074 regulatory interactions between 142 transcription factors and 3420 target genes, was developed by Mark Gerstein his collaborators (Nicholas M. Luscombe, et al, 2004), one of the most complete so far. 409 out of 792 cell cycle genes were extracted from the network and drawn into three sub-networks using Pajek (http://vlado.fmf.uni-lj.si/pub/networks/pajek/). Fig 3.8 to Fig 3.10 show sub-networks correspond to G1/S, G2/M and M/G1 transitions, with big green nodes and red nodes representing the transcription factors and target genes respectively. The arrows indicate the direction of information flow. Yellow nodes were added to indicate regulators of the transcription factors.

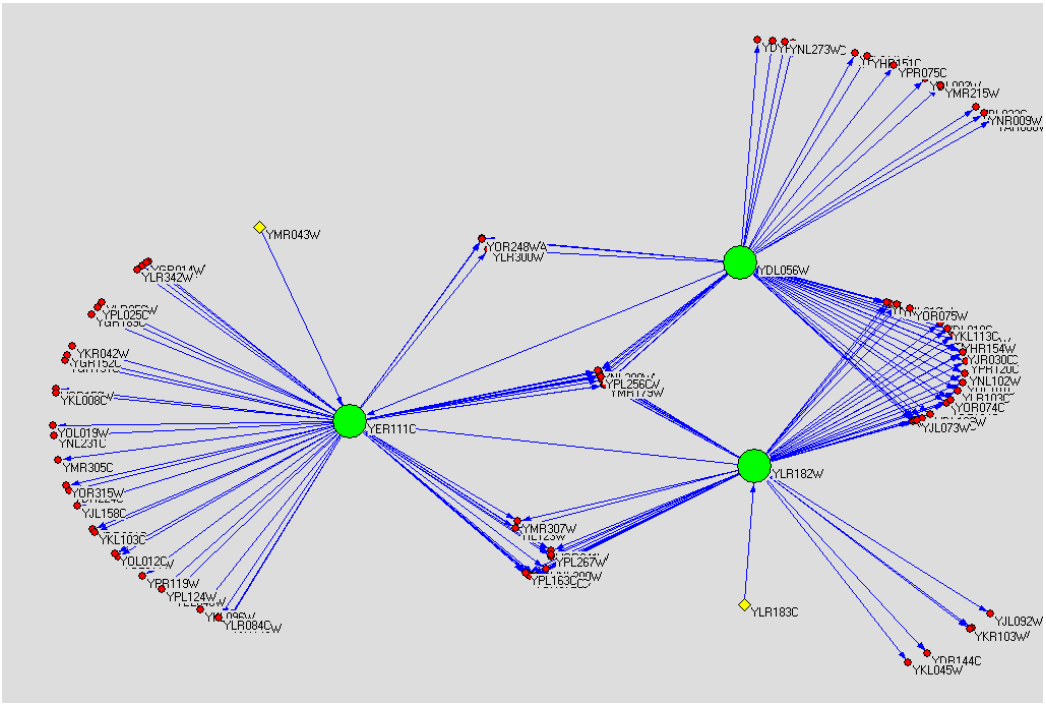

Fig 3.8: G1/S sub-network
(TF: Mbp1(YDL056W), Swi4(YER111C) and Swi6(YLR182W))

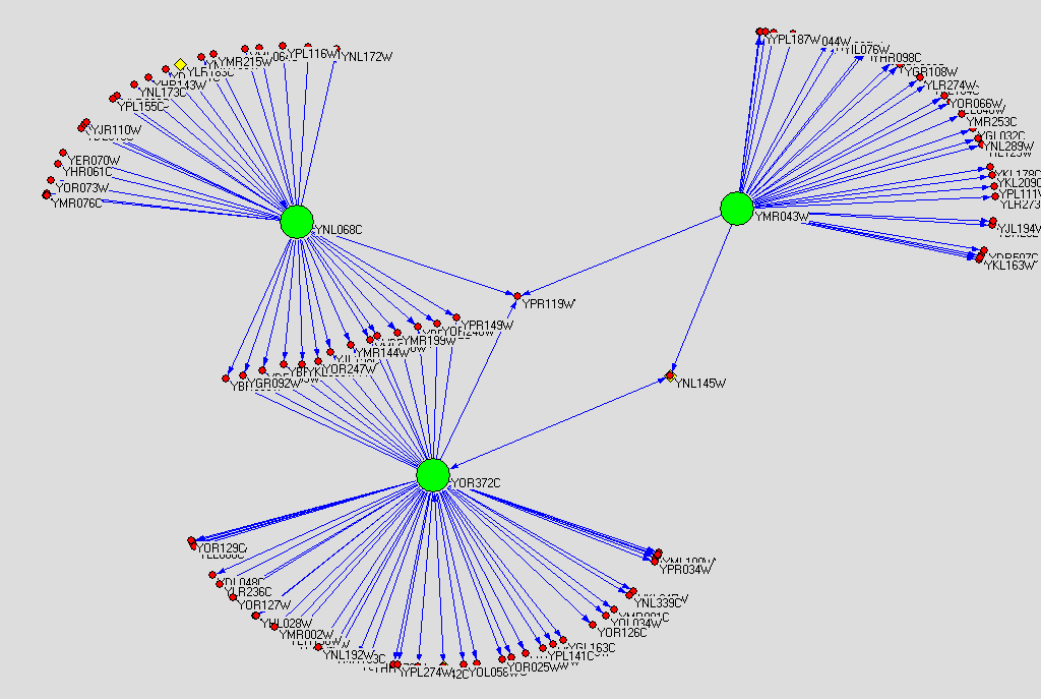

Fig 3.9: G2/M sub-network
(TF: Fkh2(YNL068C), Ndd1(YOR372C) and Mcm1(YMR043W))

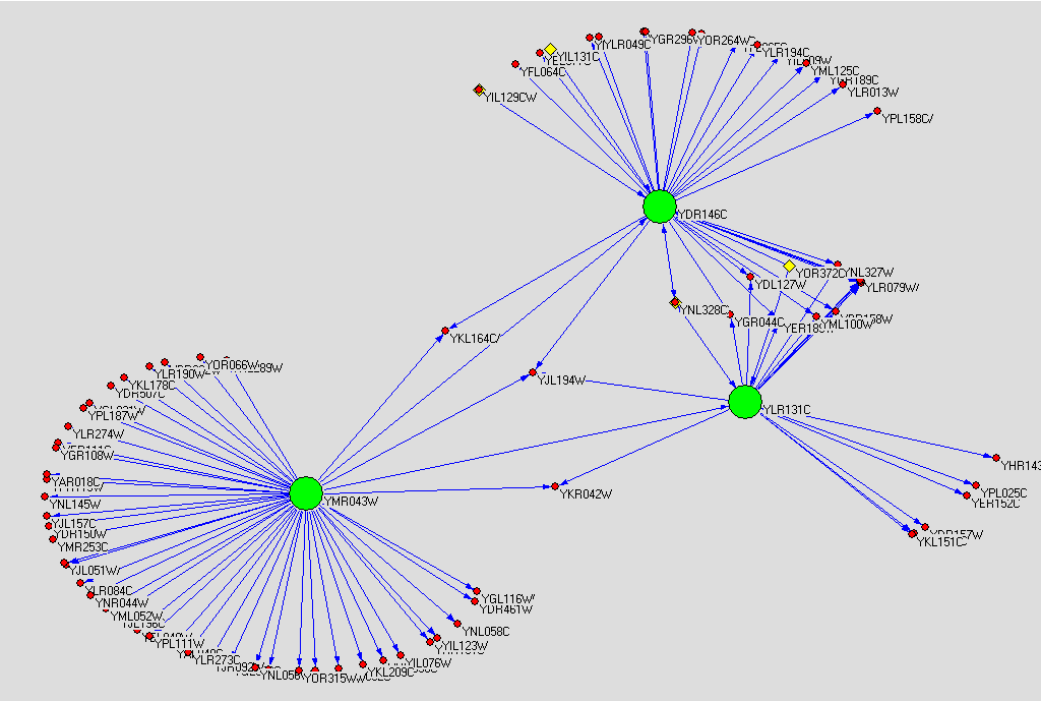

Fig 3.10: M/G1 sub-network
(TF: Mcm1(YMR043W), Swi5(YDR146C) and Ace2(YLR131C))

2.3 Cluster

Not all of the interactions between transcription factors and genes contained in the Gerstein's network have been experimental verified. Therefore, comparing the regulating network with time course micro-array data will not only serve to confirm the proposed genetic interactions, but also yield information on the execution of the regulatory program in a dynamic context.

2.3.1 G1/S Cluster

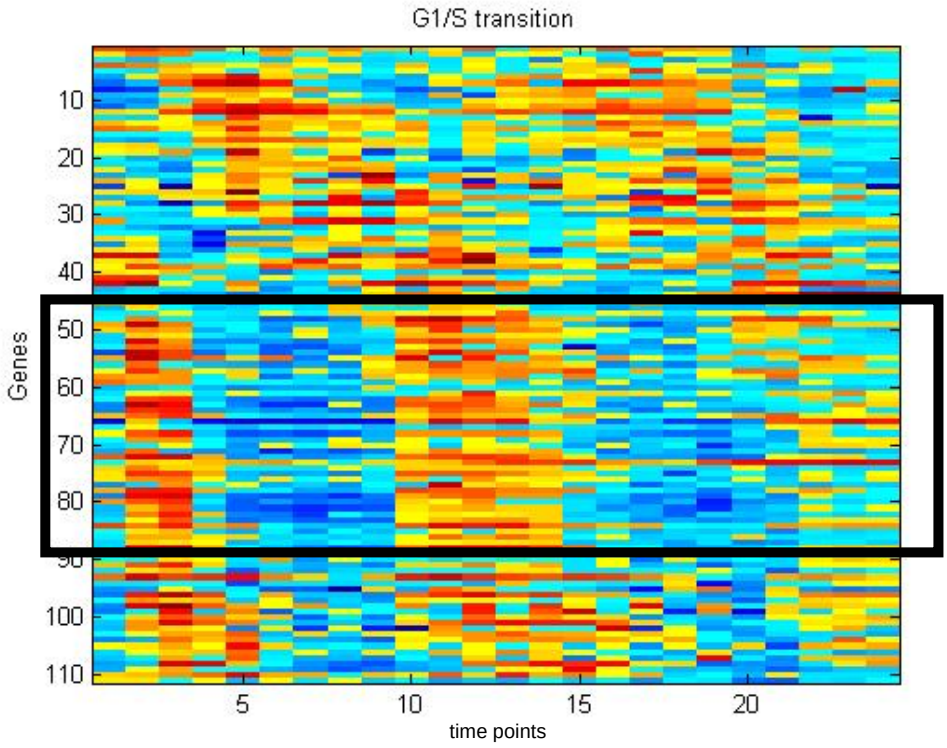

Fig 3.11: Expression of genes in G1/S sub-network

The cluster of gene 47 to 90 is clearly visible

Fig 3.11 shows gene expression controlled by G1/S transcription factors according to Gerstein's work. From the phase sorted expression pattern, we find that gene 47 to 90, form a synchronized cluster and hence are good candidate for targets of the three G1/S transcription factors alone. The other genes behavior differently could

be corregulated by other transcription factors. Checked with Richard Young's cell cycle phase description, genes in this cluster are almost exclusively 'g1' genes, while the others are categorized into 's', 'g2/m' or 'm/g1'. To verify our guess, we searched the binding motifs of several genes (http://jura.wi.mit.edu/fraenkel/regcode/).

Fig 3.12 shows the binding motifs of 'YJL158C' (No12 gene in Fig 3.11) YHR061C (No13 gene in Fig 3.11) and 'YNR044W' (No45 gene in Fig 3.11) that have different oscillation properties from the 'cluster'. The result indicates that 'YJL158C' seems to have no binding site in the promoter region. 'YHR061C' has two binding sites for fkh2 that function in next stage G2/M. 'YNR044W' could not be regulated by G1/S transcription factors at all.

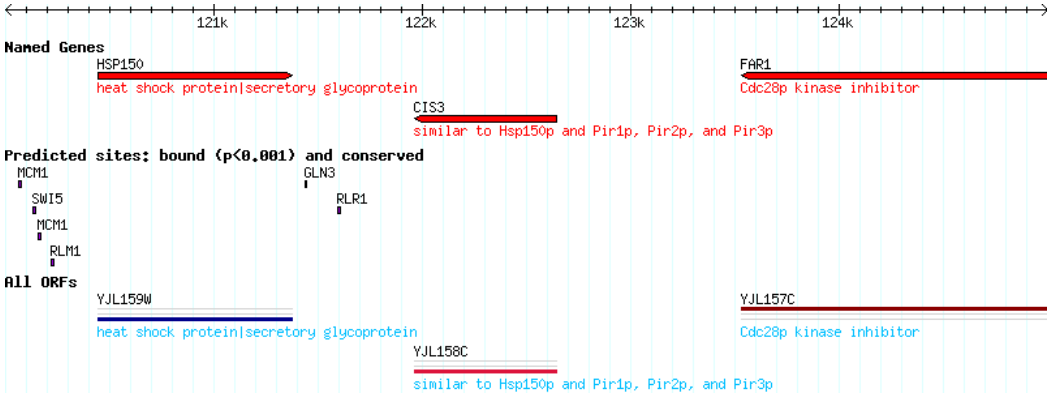

Fig 3.12_A: YJL158C

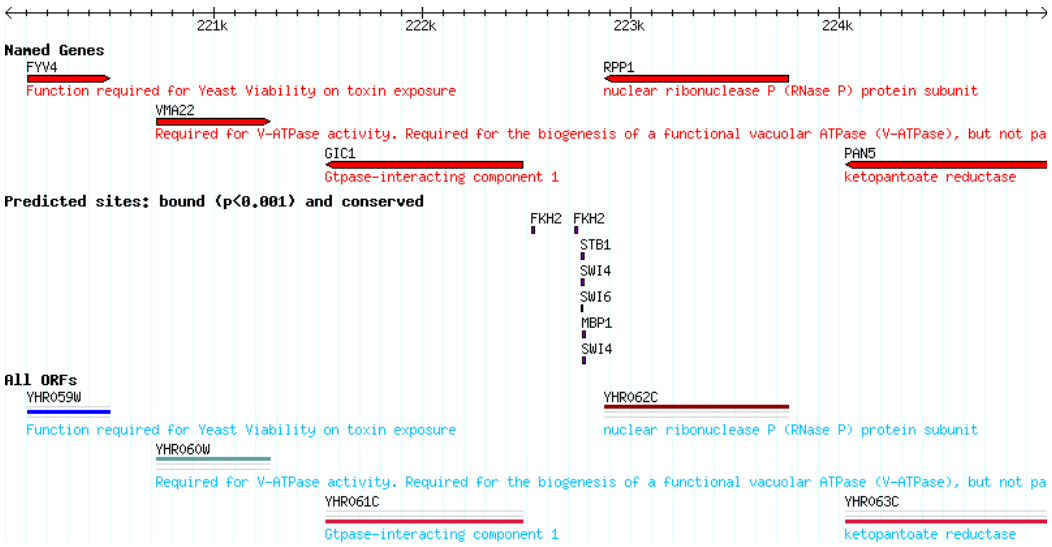

Fig 3.12_B: YHR061C

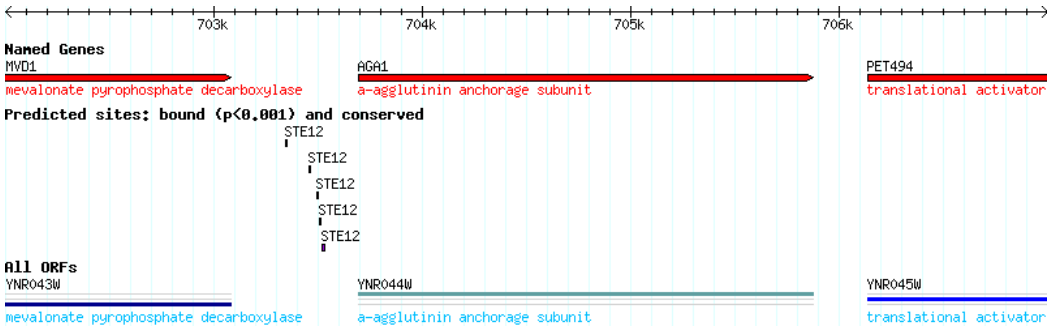

Fig 3.12_C: YNR044W

Fig 3.13 shows the binding motifs of 'YBR071W' (No49 gene in Fig 3.11), 'YJL073W' (No61 gene in Fig 3.11) and 'YDL003W' (No80 gene in Fig 3.11) in the 'cluster'. All have at least one binding site for G1/S transcription factors.

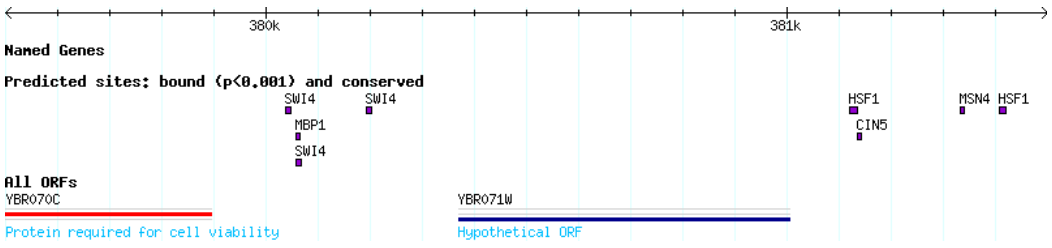

Fig 3.13_A: YBR071W

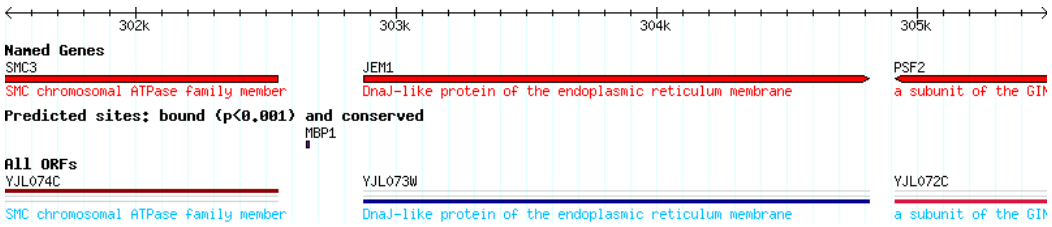

Fig 3.13_B: YJL073W

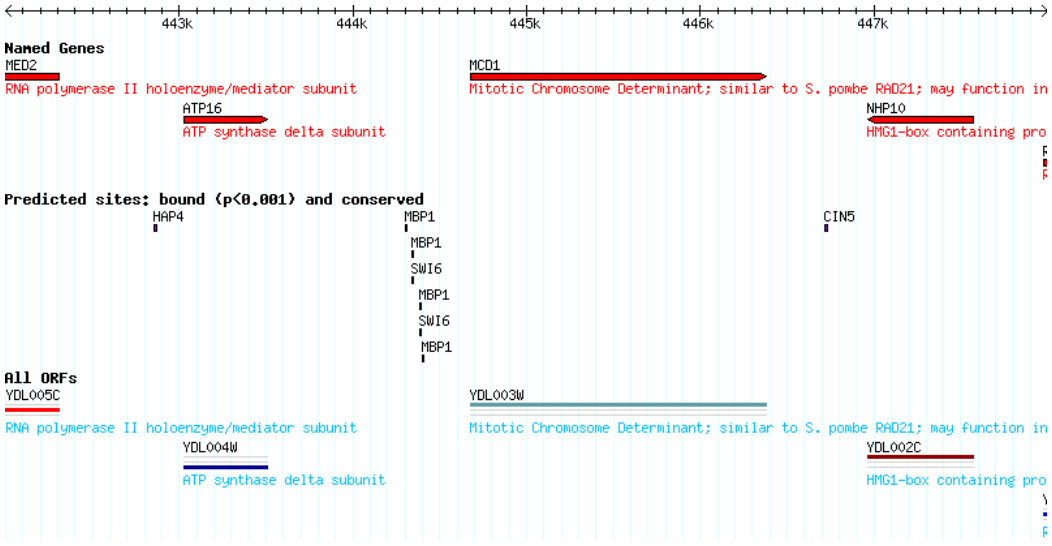

Fig 3.13_C: YDL003W

We have checked against the binding data for all the genes in Fig 3.11 when available. The sample genes shown here are representative of the situation for the majority of genes in this group. Their functional annotation is given in table 3.4.

ORF No	ORF	Young's phase	Annotation
47	YNL273W	g1	TOF1 topoisomerase I interacting factor 1
48	YOR315W	g2/m	N/A
49	YBR071W	g1	N/A
50	YDL127W	g1	PCL2 cyclin, G1/S-specific
51	YDL102W	g1	CDC2 DNA-directed DNA polymerase delta, catalytic 125 KD subunit
52	YJR030C	g1	N/A
53	YMR179W	g1	SPT21 required for normal transcription at a number of loci
54	YAR008W	g1	SEN34 tRNA splicing endonuclease gamma subunit
55	YGR109C	g1	CLB6 cyclin, B-type
56	YDR309C	g1	GIC2 Cdc42 GTPase-binding protein
57	YGR041W	g1	BUD9 budding protein
58	YGL038C	g1	OCH1 alpha-1,6-mannosyltransferase

59	YBR070C	g1	SAT2 osmotolerance protein
60	YDR279W	g1	N/A
61	YJL073W	g1	N/A
62	YNL289W	g1	PCL1 cyclin, G1/S-specific
63	YKL045W	g1	PRI2 DNA-directed DNA polymerase alpha , 58 KD subunit (DNA primase)
64	YPL267W	g1	N/A
65	YGR221C	g1	TOS2 Target of SBF, localizes to the bud neck and bud tip
66	YGR153W	g1	N/A
67	YLR286C	g1	CTS1 endochitinase
68	YOR248W	g1	N/A
69	YDR097C	g1	MSH6 DNA mismatch repair protein
70	YOR075W	g1	UFE1 syntaxin (T-SNARE) of the ER
71	YGR152C	g1	RSR1 GTP-binding protein
72	YDR528W	g1	N/A
73	YGR189C	g1	CRH1 family of putative glycosidases might exert a common role in cell wall organization
74	YML100W	g1	TSL1 alpha,alpha-trehalose-phosphate synthase, 123 KD subunit
75	YHR149C	g1	SKG6 similarity to hypothetical protein YGR221c
76	YPR120C	g1	CLB5 cyclin, B-type
77	YNL231C	g1	PDR16 protein involved in lipid biosynthesis and multidrug resistance
78	YDR501W	g1	PLM2 PLasmid Maintenance mutant shows 2mu-m plasmid instability
79	YOL019W	g1	N/A
80	YDL003W	g1	MCD1 Mitotic Chromosome Determinant
81	YGR151C	g1	N/A
82	YAR007C	g1	RFA1 DNA replication factor A, 69 KD subunit
83	YLR103C	g1	CDC45 required for minichromosome maintenance and initiation of chromosomal DNA replication
84	YDL018C	g1	N/A
85	YER095W	g1	RAD51 DNA repair protein
86	YJL074C	g1	SMC3 required for structural maintenance of chromosomes
87	YCR065W	g1	HCM1 transcription factor
88	YNL102W	g1	POL1 DNA-directed DNA polymerase alpha, 180 KD subunit
89	YPL256C	g1	CLN2 cyclin, G1/S-specific
90	YDL101C	g1	DUN1 protein kinase

Table 3.4: G1/S Cluster (Gene 47 to 90 in Fig 3.9)

In conclusion, the G1/S cluster contains 44 genes. The well studied G1 cyclin genes Clb5,6 and Cln2 are included as well as budding protein BUD9. Others with similar behavior in the experiment were added to form a cluster of G1/S coregulated genes.

2.3.2 G2/M

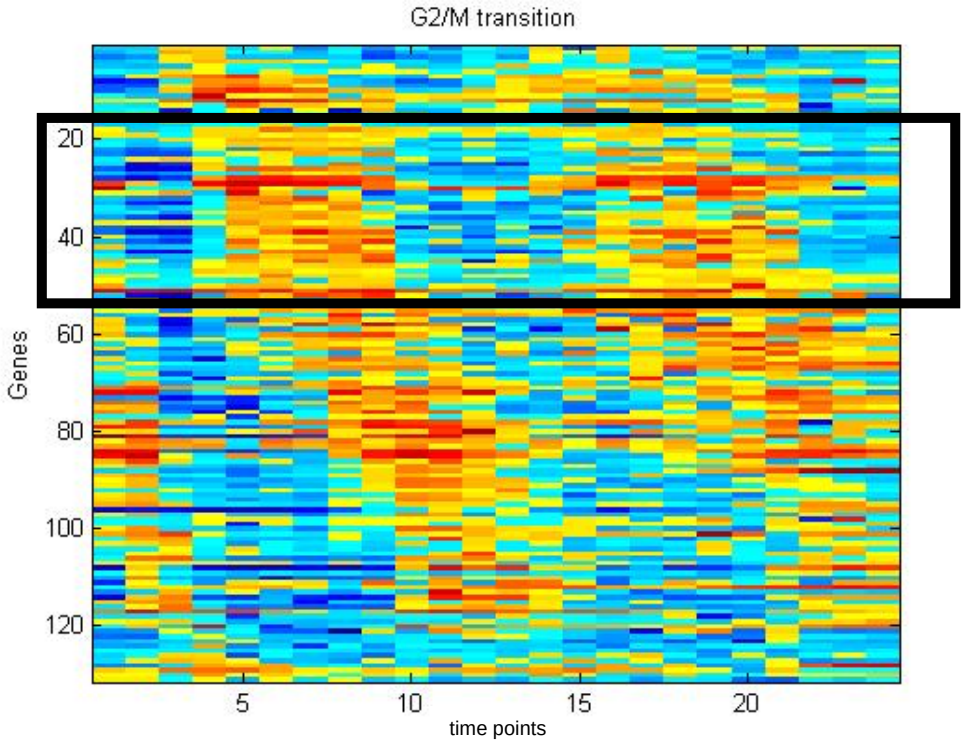

Fig 3.14: Expression of genes in G2/M sub-network

The cluster of gene 17 to 56 is clearly visible

Fig 3.14 shows gene expression controlled by G2/M transcription factors according to Gerstein's work. Similarly, we believe that gene 17 to 56 is within the same cluster that is most possibly controlled by M/G1 transcription factors. We also checked with Richard Young's cell cycle phase description and genes in the 'cluster' are almost 'g2/m' genes, while the others do not have such a regulation. However, there seems to be a small cluster between gene 70 and 90 regulated by Mcm1 in M/G1.

ORF No	ORF	Young's phase	Annotation
17	YML064C	g2/m	TEM1 GTP-binding protein of the RAS superfamily
18	YDR150W	s/g2	NUM1 nuclear migration protein
19	YPL111W	g2/m	CAR1 arginase
20	YOR247W	g1	N/A
21	YPL155C	s/g2	KIP2 kinesin-related protein
22	YBR133C	s/g2	HSL7 adapter in a regulatory pathway that relieves tyrosine phosphorylation of Cdc28
23	YIL158W	g2/m	N/A
24	YGL255W	g2/m	ZRT1 Zinc transporter I
25	YNL192W	m/g1	CHS1 chitin synthase I
26	YMR001C	g2/m	CDC5 protein kinase, involved in regulation of DNA replication
27	YML052W	g2/m	N/A
28	YLR131C	g2/m	ACE2 metallothionein expression activator
29	YHL028W	g2/m	WSC4 Cell wall integrity and stress response component 4
30	YGR108W	g2/m	CLB1 cyclin, G2/M-specific
31	YAL053W	g1	N/A
32	YKL096W-A	s/g2	CWP2 cell wall mannoprotein
33	YPL141C	s/g2	N/A
34	YOR025W	g2/m	HST3 silencing protein
35	YMR144W	s	N/A
36	YLR084C	g2/m	N/A
37	YDR146C	g2/m	SWI5 transcription factor
38	YMR002W	s/g2	N/A
39	YBR138C	g2/m	HDR1 High-Dosage Reductional segregation defective
40	YPR149W	g2/m	NCE102 involved in non-classical protein export pathway
41	YPR119W	g2/m	CLB2 cyclin, G2/M-specific
42	YGL008C	g2/m	PMA1 H+-transporting P-type ATPase
43	YOR129C	g2/m	N/A
44	YLR190W	g2/m	MMR1 protein localized to bud sites and tips, mother-bud junction
45	YCR024C-A	g2/m	PMP1 H+-ATPase subunit, plasma membrane
46	YHR151C	g2/m	N/A
47	YGL162W	g2/m	SUT1 hypoxic protein involved in sterol uptake
48	YNL172W	g2/m	APC1 subunit of anaphase-promoting complex (cyclosome)
49	YOR023C	g2/m	AHC1 component of the ADA histone acetyltransferase complex
50	YNL056W	g2/m	N/A
51	YJR092W	g2/m	BUD4 budding protein
52	YDL048C	g2/m	STP4 involved in pre-tRNA splicing and in uptake of branched-chain amino acids
53	YBR038W	g2/m	CHS2 chitin synthase II
54	YER070W	g1	RNR1 ribonucleoside-diphosphate reductase, large subunit
55	YBR092C	g2/m	PHO3 constitutive acid phosphatase precursor
56	YMR253C	g2/m	N/A

Table 3.5: G2/M Cluster (Gene 17 to 56 in Fig 3.13)

In conclusion, the G2/M cluster contains 40 genes. The well studied G2/M cyclin genes Clb1,2 are involved and others with similar behavior in the experiment were added to form a cluster of G2/M coregulated genes.

2.3.3 M/G1

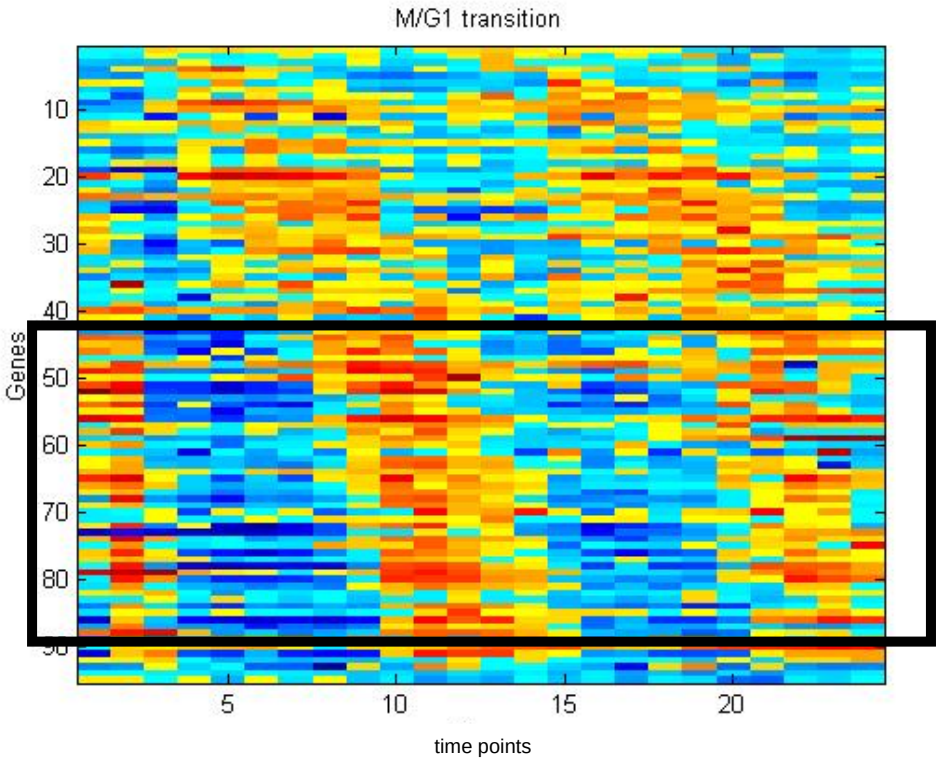

Fig 3.15: Expression of genes in M/G1 sub-network

The cluster of gene 43 to 90 is clearly visible

Fig 3.15 shows gene expression controlled by M/G1 transcription factors according to Gerstein's work. Similarly, we believe that gene 43 to 90 is within the same cluster that is most possibly controlled by M/G1 transcription factors. We also checked with Richard Young's cell cycle phase description. Genes in the 'cluster' are almost exclusively 'm/g1' or 'g1' genes, while the others do not have such a

regulation. However, there also seems to be a small cluster between gene 20 and 30 which is regulated by Mcm1 in G2/M.

ORF No	ORF	Young's phase	Annotation
43	YJL194W	m/g1	N/A
44	YKL178C	m/g1	MMR1 protein localized to bud sites and tips, mother-bud junction
45	YOR066W	m/g1	PMP1 H+-ATPase subunit, plasma membrane
46	YLR274W	m/g1	N/A
47	YPL187W	m/g1	SUT1 hypoxic protein involved in sterol uptake
48	YBR202W	g2/m	APC1 subunit of anaphase-promoting complex (cyclosome)
49	YLR273C	m/g1	AHC1 component of the ADA histone acetyltransferase complex
50	YJL159W	m/g1	N/A
51	YJL196C	m/g1	BUD4 budding protein
52	YKL185W	m/g1	STP4 involved in pre-tRNA splicing and in uptake of branched-chain amino acids
53	YKL164C	m/g1	CHS2 chitin synthase II
54	YNL328C	m/g1	RNR1 ribonucleoside-diphosphate reductase, large subunit
55	YPL158C	m/g1	PHO3 constitutive acid phosphatase precursor
56	YIL009W	m/g1	N/A
57	YKL163W	m/g1	PDS5 precocious dissociation of sister chromatids
58	YGR234W	m/g1	CDC20 cell division control protein
59	YDR055W	m/g1	MRH1 membrane protein related to Hsp30p
60	YNR044W	m/g1	IAH1 isoamyl acetate hydrolytic enzyme
61	YEL040W	m/g1	N/A
62	YER189W	g1	STE2 pheromone alpha-factor receptor
63	YFL064C	g1	KIN3 ser/thr protein kinase
64	YER152C	g1	AGA2 a-agglutinin binding subunit
65	YLR079W	m/g1	DBF2 ser/thr protein kinase related to Dbf20p
66	YGR086C	m/g1	MFA2 mating pheromone a-factor 2
67	YEL077C	g1	BNS1 Bypasses Need for Spo12p
68	YBR158W	m/g1	RGA1 RHO-type GTPase-activating protein for Cdc42p
69	YER190W	g1	SPO12 sporulation protein
70	YKL151C	m/g1	SWI4 transcription factor
71	YOR315W	g2/m	CLN3 cyclin, G1/S-specific
72	YNL327W	m/g1	CDC6 cell division control protein
73	YGL089C	g1	STE3 pheromone a-factor receptor
74	YDL127W	g1	N/A
75	YPL283C	m/g1	CDC46 cell division control protein
76	YLR049C	g1	MF(ALPHA)1 mating pheromone alpha-1 precursor
77	YGR296W	g1	CDC47 cell division control protein
78	YGL028C	g1	PIG1 putative type 1 phosphatase regulatory subunit
79	YLR194C	m/g1	HSP150 member of the Pir1p/Hsp150p/Pir3p family
80	YGR044C	g1	ELO1 fatty acid elongation protein
81	YGR041W	g1	PIR1 required for tolerance to heat shock
82	YJL157C	g2/m	CLN1 cyclin, G1/S-specific
83	YJL051W	g2/m	KEL2 involved in cell fusion and morphogenesis
84	YJL078C	g1	PIR3 member of the Pir1p/Pir2p/Pir3p family

85	YNL289W	g1	EXG2 exo-beta-1,3-glucanase minor isoform
86	YLR286C	g1	N/A
87	YDR528W	g1	N/A
88	YGR189C	g1	AGA1 a-agglutinin anchor subunit
89	YDR461W	g1	UTR2 cell wall protein

Table 3.6: M/G1 Cluster (Gene 43 to 90 in Fig 3.14)

In conclusion, the M/G1 cluster contains 48 genes. Cdc20, Cdc6 and Cln3 are involved where Cdc20 and Cdc6 trigger the exit from mitosis and Cln3 is the cell cycle signal for cell cycle division. Others with similar behavior in the experiment were added to form a cluster of M/G1 coregulated genes.

These clusters provide a foundation for understanding the transcriptional mechanism of cell cycle regulation. Fig 3.16 shows the entire cluster-gram of the transcription factor target genes. The corresponding network is shown in Fig 3.17. As discussed previously, these identified corregulated genes share common binding sites. Presumably the genes within the same group somehow have related functions. The 44 G1/S cluster genes include CLN2, CLB5,6, CDC2, PRI2, SEN34, RFA1, CDC45 and many other genes involved in DNA replication. Many genes known to be involved in mitosis is found in G2/M cluster which contains 40 genes, such as CDC5, CLB1,2, SWI5, APC1 and BUD4. The G2/M cluster which contains 48 genes includes 4 cell division control proteins CDC20, CDC6, CDC46 and CDC47.

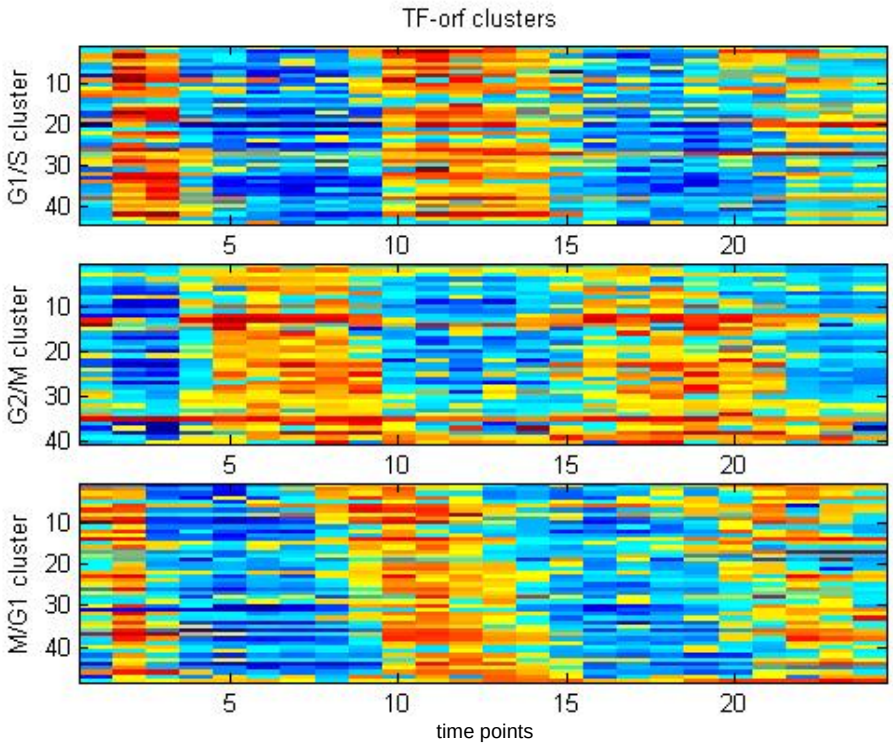

Fig 3.16: the clusters of cell cycle stage specific TF_orf groups

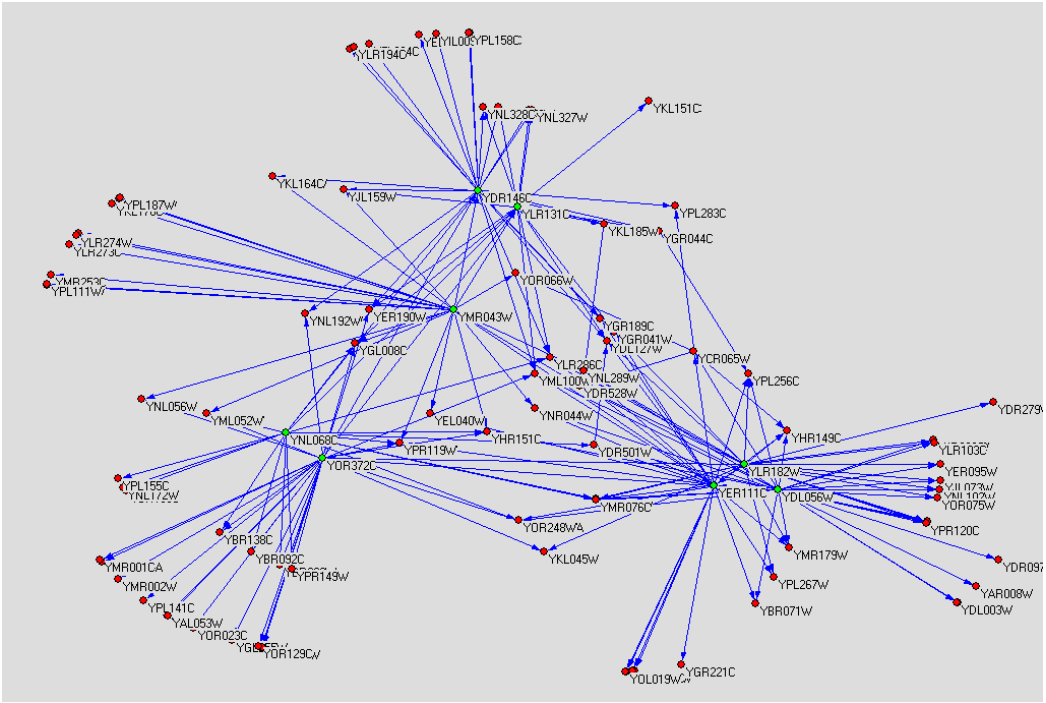

Fig 3.17: Simplified cell cycle TF_ orf network

Chapter 4

Achievements and Further Work

The purpose of the present project is to study the regulatory program of yeast cell cycle. Implementation of Prof. Tang Chao's model gives us a good demonstration of cell cycle's robustness and stability. By digging into gene expression program and further grouping the genes into phase synchronized clusters, we confirmed the proposed genetic interactions.

Even though the cell cycle regulatory proteins are few and their roles well-characterized, the execution of the dynamic program is rather complex and many of the details are yet to be understood. In the thesis, we have only examined the gene expression data from one time course experiment. With more data under various experimental conditions, along with good binding data, one may overcome some of the intrinsic issues with noise and obtain a more complete picture of cell cycle gene regulation. Furthermore, beyond the static interactions between the transcription factors and target genes, we would like to understand in more detail the turning on/off process of a given gene by one or more transcription factors, and how such processes at the single gene level are fine tuned to meet the global demands of cell growth, replication, and division.

Bibliography

- Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K. & Walter, P. *Molecular Biology of the Cell*, Fourth Edition (2002).
- Iyer, V. R. et al. Genomic binding sites of the yeast cell-cycle transcription factors SBF and MBF. *Nature* 409, 533–538 (2001).
- Lee, T. I. et al. Transcriptional regulatory networks in Saccharomyces cerevisiae. *Science* 298, 799–804 (2002).
- Wang, W., Cherry, J.M., Botstein, D. & Li, H. A systematic approach to reconstructing transcription networks in Saccharomyces cerevisiae. *Proc. Natl. Acad. Sci.* **99**, 16893-16898(2002).
- Christopher T. Harbison, et al, Transcriptional regulatory code of a eukaryotic genome, *Nature*, **431**, 99-104, (2004)
- Fangting Li, Tao Long, Ying Lu, Qi Ouyang, and Chao Tang, The yeast cell-cycle network is robustly designed, *PNAS*, **101**, 14, 4781-4786, (2004)
- Nicolas E. Buchler, Ulrich Gerland, and Terence Hwa, On schemes of combinatorial transcription logic, *PNAS*, **100**, 9, 5136-5141, (2003)
- Paul T. Spellman, et al, Comprehensive Identification of Cell Cycle-regulated Genes of the Yeast *Saccharomyces cerevisiae* by Microarray Hybridization, *Molecular Biology of the Cell*, **9**, 3273-3297, (1998)
- Bruce Futcher, Transcriptional regulatory networks and the yeast cell cycle, *Current Opinion in Cell Biology*, **14**, 676-683, (2002)
- Itamar Simon, et al, Serial Regulation of Transcriptional Regulators in the Yeast Cell Cycle, *Cell*, **106**, 697-708, (2001)

- Gordon Chua, et al, Transcriptional networks: reverse-engineering gene regulation on a global scale, *Current Opinion in Microbiology*, **7**: 638-646, (2004)
- Ian A. Taylor, et al, Characterization of the DNA-Binding Domains from the Yeast Cell-Cycle Transcription Factors Mbp1 and Swi4, *Biochemistry*, **39**, 3943-3954, (2000)
- Dien BS, Srienc F, Bromodeoxyuridine labeling and flow cytometric identification of replicating Saccharomyces cerevisiae cells: lengths of cell cycle phases and population variability at specific cell cycle positions, *Biotechnol Prog*, **7(4):291-8**, (1991)
- C Wittenberg, Cell cycle: Cyclin guides the way, *Nature* **434**, 34 - 35 (2005).
- Mart Loog and David O. Morgan, Cyclin specificity in the phosphorylation of cyclin-dependent kinase substrates, *Nature* **434**, 104-108 (2005)
- Alberghina et al, A cell sizer network involving Cln3 and Far1 controls entrance into S phase in the mitotic cycle of budding yeast, *J. Cell Biol,* **167**, 433-443.(2004)
- Nash, R., Tokiwa, G., Anand, S., Erickson, K. and Futcher, A.B, The WHI1+ gene of *Saccharomyces cerevisiae* tethers cell division to cell size and is a cyclin homolog. *EMBO J.* **7**, 4335-4346, (1988).
- Tyers, M., Tokiwa, G. and Futcher B, Comparison of the *Saccharomyces cerevisiae* G1 cyclins: Cln3 may be an upstream activator of Cln1, Cln2 and other cyclins. *EMBO J.* **12**:1955-68, (1993)
- Schwob, E. and Nasmyth, K, CLB5 and CLB6, a new pair of B cyclins involved in DNA replication in *Saccharomyces cerevisiae*. *Genes Dev*. **7**:1160-1175, (1993).

- Nicholas M. Luscombe, et al, Genomic analysis of regulatory network dynamics reveals large topological changes, *Nature*, **431,** 308-312, (2004)

Appendix -- C programming code for cell cycle model implementation

```
//          Implementation of Chao's model used to investigate the fixed points and biological pathway
under various dynamic rules
//          Andy Cai
//          11/2004

#include <stdlib.h>
#include <stdio.h>
#include <math.h>

void main()
{
        int option, option1, option2, option3;
        int Sn[] = {0,0,0,0,0,0,0,0,0,0,0};
        int So[] = {0,0,0,0,0,0,0,0,0,0,0};
        int Temp[] = {0,0,0,0,0,0,0,0,0,0,0};
        int Sum[] = {0,0,0,0,0,0,0,0,0,0,0};
        int S[] =  {0,0,0,0,0,0,0,0,0,0,0};
        int Selfdeg[] = {1,1,1,1,1};
        int Basinsize[] = {0,0,0,0,0,1,0};
        int i, j, count, firstentry;
        int Ag, Ar, td;
        int timeinterval;

        int PF1[] = {0,0,0,0,1,0,0,0,1,0,0};
        int PF2[] = {0,0,1,1,0,0,0,0,0,0,0};
        int PF3[] = {0,1,0,0,1,0,0,0,1,0,0};
        int PF4[] = {0,0,0,0,0,0,0,0,1,0,0};
        int PF5[] = {0,1,0,0,0,0,0,0,1,0,0};
        int PF6[] = {0,0,0,0,0,0,0,0,0,0,0};
        int PF7[] = {0,0,0,0,1,0,0,0,0,0,0};

        printf("To investigate the trajectories in state space; please enter 1. \n");
        printf("To do statistics of the big fixed points of the cell-cycle network; please enter 2. \n");
        printf("To quit the program; please enter 3. \n");
        printf("Please select one option: ");
        scanf("%d", &option);

        while(option != 3)
        {
                for(i=0;i<=10;i++)
                {
                        Sn[i] = So[i] = Temp[i] = Sum[i] = S[i] = 0;
                }

                for(i=0;i<=4;i++)
                {
                        Selfdeg[i] = 1;
                }

                for(i=0;i<=6;i++)
                {
                        Basinsize[i] = 0;
                        Basinsize[5] = 1;
                }

                if (option == 1)
                {
                        printf("\nBack to main list, please enter 0; to continue, please press 1: ");
                        scanf("%d", &option1);

                        while(option1 != 0)
                        {
                                for(i=0;i<=10;i++)
```

```
{
	Sn[i] = So[i] = Temp[i] = Sum[i] = S[i] = 0;
}

printf("\nPlease assign values for Ag, Ar and td(Note: The value of Ar must larger or equal to that of Ag). \n");
	scanf("%d%d%d", &Ag, &Ar, &td);

	printf("Ag = %d\t Ar = %d\t td = %d\n", Ag, Ar, td);

	for(i=0;i<=10;i++)
	{
		Sum[i] = 0;
	}

	for(i=0;i<=4;i++)
	{
		Selfdeg[i] = 1;
	}

firstentry = 1;
printf("\nPlease assign state statuses for 11 nodes(1 for activated node; 0 for deactivated node), Cln3, MBF, SBF, Cln1,2, Cdh1, Swi5, Cdc20&Cdc14, Clb5,6, Sic1, Clb1,2 and Mcm1&SFF: \n");
scanf("%d%d%d%d%d%d%d%d%d%d%d", &So[0], &So[1], &So[2], &So[3], &So[4], &So[5], &So[6], &So[7], &So[8], &So[9], &So[10]);


	printf("Cln3\tMBF\tSBF\tCln1,2\tCdh1\tSwi5\tCdc20&Cdc14\tClb5,6\tSic1\tClb1,2\tMcm1&SFF \n");
printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\n", So[0], So[1], So[2], So[3], So[4], So[5], So[6], So[7], So[8], So[9], So[10]);

timeinterval = 0;

while(timeinterval!=td)
	{
	for(i=0;i<=10;i++)
	{
		Sum[i] = 0;
	}

	if (firstentry == 1)
	{
		for(i=0;i<=10;i++)
		{
			Sn[i] = So[i];
		}
		firstentry = 0;
	}

	if (Sn[0] == 1)//self-degradation
	{
		Sum[1] = Sum[1] + Ag;
		Sum[2] = Sum[2] + Ag;

		if(Selfdeg[0] == td)
			Sum[0] = -1;
		else Selfdeg[0]++;
	}

	if (Sn[1] == 1)
	{
		Sum[7] = Sum[7] + Ag;
	}

	if (Sn[2] == 1)
```

```
{
        Sum[3] = Sum[3] + Ag;
}

if (Sn[3] == 1)//self=degradation
{
        if (Sn[2] == 0)
        {
                Sum[4] = Sum[4] - Ar;
                Sum[8] = Sum[8] - Ar;

                if(Selfdeg[1] == td)
                        Sum[3] = -1;
                else Selfdeg[1]++;
        }

        else if(Sn[2] == 1)
        {
                Sum[4] = Sum[4] - Ar;
                Sum[8] = Sum[8] - Ar;
                Selfdeg[1] = 1;
        }
}

if (Sn[4] == 1)
{
        Sum[9] = Sum[9] - Ar;
}

if (Sn[5] == 1)//self-degradation
{
        if (Sn[6] + Sn[10] - Sn[9] == 0)
        {
                Sum[8] = Sum[8] + Ag;

                if(Selfdeg[2] == td)
                        Sum[5] = -1;
                else Selfdeg[2]++;
        }

        else if(Sn[6] + Sn[10] - Sn[9] > 0)
        {
                Sum[8] = Sum[8] + Ag;
                Selfdeg[2] = 1;
        }
}

if (Sn[6] == 1)//self-degradation
{
        if (Sn[9] + Sn[10] == 0)
        {
                Sum[4] = Sum[4] + Ag;
                Sum[5] = Sum[5] + Ag;
                Sum[8] = Sum[8] + Ag;
                Sum[7] = Sum[7] - Ar;
                Sum[9] = Sum[9] - Ar;

                if(Selfdeg[3] == td)
                        Sum[6] = -1;
                else Selfdeg[3]++;
        }

        else if (Sn[9] + Sn[10] > 0)
        {
                Sum[4] = Sum[4] + Ag;
                Sum[5] = Sum[5] + Ag;
                Sum[8] = Sum[8] + Ag;
```

```
                        Sum[7] = Sum[7] - Ar;
                        Sum[9] = Sum[9] - Ar;
                        Selfdeg[3] = 1;
                }
}

if (Sn[7] == 1)
{
        Sum[9] = Sum[9] + Ag;
        Sum[10] = Sum[10] + Ag;
        Sum[4] = Sum[4] - Ar;
        Sum[8] = Sum[8] - Ar;
}

if (Sn[8] == 1)
{
        Sum[7] = Sum[7] - Ar;
        Sum[9] = Sum[9] - Ar;
}

if (Sn[9] == 1)
{
        Sum[6] = Sum[6] + Ag;
        Sum[10] = Sum[10] + Ag;
        Sum[1] = Sum[1] - Ar;
        Sum[2] = Sum[2] - Ar;
        Sum[4] = Sum[4] - Ar;
        Sum[5] = Sum[5] - Ar;
        Sum[8] = Sum[8] - Ar;
}

if (Sn[10] == 1)//self-degradation
{
        if (Sn[9] == 0)
        {
                Sum[5] = Sum[5] + Ag;
                Sum[6] = Sum[6] + Ag;
                Sum[9] = Sum[9] + Ag;
                if(Selfdeg[4] == td)
                        Sum[10] = -1;
                else Selfdeg[4]++;
        }

        else if(Sn[9] == 1)
        {
                Sum[5] = Sum[5] + Ag;
                Sum[6] = Sum[6] + Ag;
                Sum[9] = Sum[9] + Ag;
        }
}
//update the value for status conditions for all nodes

for(i=0;i<=10;i++)
        {
                Temp[i] = Sn[i];
        }//Store the old statuses in a Temp array

for(i=0;i<=10;i++)
        {
                So[i] = Temp[i];
        }//update the old statuses for 11 nodes

for(i=0;i<=10;i++)
{
        if (Sum[i] > 0)
                Sn[i] = 1;
        else if (Sum[i] < 0)
```

```
                                                Sn[i] = 0;
                                        else Sn[i] = So[i];
                                }//The new statuses of all 11 nodes

                                if(Sn[0] == So[0] && Sn[1] == So[1] && Sn[2] == So[2] &&
Sn[3] == So[3] && Sn[4] == So[4] && Sn[5] == So[5] && Sn[6] == So[6] && Sn[7] == So[7] && Sn[8] ==
So[8] && Sn[9] == So[9] && Sn[10] == So[10])
                                {
                                        timeinterval++;
                                }

                                else timeinterval = 0;

                                printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t\t%d\t%d\t%d\t%d\n",
Sn[0], Sn[1], Sn[2], Sn[3], Sn[4], Sn[5], Sn[6], Sn[7], Sn[8], Sn[9], Sn[10]);

                                }//end of while loop for the fixed point

                        printf("\nBack to main list, please enter 0; to continue, please press 1: ");
                        scanf("%d", &option1);
                        }//end of the while loop for option1

                }//case for option 1

                else if (option == 2)
                {
                        printf("\nBack to main list, please enter 0; to continue, please press 1: ");
                        scanf("%d", &option2);

                        while(option2 != 0)
                        {
                                for(i=0;i<=10;i++)
                                {
                                Sn[i] = So[i] = Temp[i] = Sum[i] = S[i] = 0;
                                }

                                for(i=0;i<=4;i++)
                                {
                                        Selfdeg[i] = 1;
                                }

                                for(i=0;i<=6;i++)
                                {
                                        Basinsize[i] = 0;
                                        Basinsize[5] = 1;
                                }

                                printf("\nTo do statistics with default dynamic rule settings, please
enter 1. \n");
                                printf("To do statistics with other dynamics rule settings, please
enter 2. \n");
                                printf("Please make a choice: ");
                                scanf("%d", &option3);

                                if (option3 == 1)
                                {
                                        for(count=1;count<=2048;count++)
                                        {
                                                for(j=0;j<=10;j++)
                                                {
                                                        if((count%(int)pow(2,j+1))<(int)pow(2,j))
                                                                Sn[j] = 0;
                                                        else
if((count%(int)pow(2,j+1))>=(int)pow(2,j))
                                                                Sn[j] = 1;
                                                }
```

```
while(Sn[0] != So[0] || Sn[1] != So[1] || Sn[2] != So[2] ||
Sn[3] != So[3] || Sn[4] != So[4] || Sn[5] != So[5] || Sn[6] != So[6] || Sn[7] != So[7] || Sn[8] != So[8] ||
Sn[9] != So[9] || Sn[10] != So[10])
{
    for(i=0;i<=10;i++)
    {
        Sum[i] = 0;
    }

    if (firstentry == 1)
    {
        for(i=0;i<=10;i++)
        {
            Sn[i] = So[i];
        }
        firstentry = 0;
    }

    if (Sn[0] == 1)//self-degradation
    {
        Sum[1] = Sum[1] + 1;
        Sum[2] = Sum[2] + 1;
        Sum[0] = -1;
    }

    if (Sn[1] == 1)
    {
        Sum[7] = Sum[7] + 1;
    }

    if (Sn[2] == 1)
    {
        Sum[3] = Sum[3] + 1;
    }

    if (Sn[3] == 1)//self=degradation
    {
        if (Sn[2] == 0)
        {
            Sum[4] = Sum[4] - 1;
            Sum[8] = Sum[8] - 1;
            Sum[3] = -1;
        }

        else if(Sn[2] == 1)
        {
            Sum[4] = Sum[4] - 1;
            Sum[8] = Sum[8] - 1;
        }
    }

    if (Sn[4] == 1)
    {
        Sum[9] = Sum[9] - 1;
    }

    if (Sn[5] == 1)//self-degradation
    {
        if (Sn[6] + Sn[10] - Sn[9] == 0)
        {
            Sum[8] = Sum[8] + 1;
            Sum[5] = -1;
        }

        else if(Sn[6] + Sn[10] - Sn[9] > 0)
        {
```

```
                Sum[8] = Sum[8] + 1;
        }
}

if (Sn[6] == 1)//self-degradation
{
        if (Sn[9] + Sn[10] == 0)
        {
                Sum[4] = Sum[4] + 1;
                Sum[5] = Sum[5] + 1;
                Sum[8] = Sum[8] + 1;
                Sum[7] = Sum[7] - 1;
                Sum[9] = Sum[9] - 1;
                Sum[6] = -1;
        }

        else if (Sn[9] + Sn[10] > 0)
        {
                Sum[4] = Sum[4] + 1;
                Sum[5] = Sum[5] + 1;
                Sum[8] = Sum[8] + 1;
                Sum[7] = Sum[7] - 1;
                Sum[9] = Sum[9] - 1;
        }
}

if (Sn[7] == 1)
{
        Sum[9] = Sum[9] + 1;
        Sum[10] = Sum[10] + 1;
        Sum[4] = Sum[4] - 1;
        Sum[8] = Sum[8] - 1;
}

if (Sn[8] == 1)
{
        Sum[7] = Sum[7] - 1;
        Sum[9] = Sum[9] - 1;
}

if (Sn[9] == 1)
{
        Sum[6] = Sum[6] + 1;
        Sum[10] = Sum[10] + 1;
        Sum[1] = Sum[1] - 1;
        Sum[2] = Sum[2] - 1;
        Sum[4] = Sum[4] - 1;
        Sum[5] = Sum[5] - 1;
        Sum[8] = Sum[8] - 1;
}

if (Sn[10] == 1)//self-degradation
{
        if (Sn[9] == 0)
        {
                Sum[5] = Sum[5] + 1;
                Sum[6] = Sum[6] + 1;
                Sum[9] = Sum[9] + 1;
                Sum[10] = -1;
        }

        else if(Sn[9] == 1)
        {
                Sum[5] = Sum[5] + 1;
                Sum[6] = Sum[6] + 1;
                Sum[9] = Sum[9] + 1;
        }
```

```
                                        }
                                        //update the value for status conditions for all
nodes

                                        for(i=0;i<=10;i++)
                                                {
                                                        Temp[i] = Sn[i];
                                                }//Store the old statuses in a Temp array

                                        for(i=0;i<=10;i++)
                                                {
                                                        So[i] = Temp[i];
                                                }//update the old statuses for 11 nodes

                                        for(i=0;i<=10;i++)
                                        {
                                                if (Sum[i] > 0)
                                                        Sn[i] = 1;
                                                else if (Sum[i] < 0)
                                                        Sn[i] = 0;
                                                else Sn[i] = So[i];
                                        }//The new statuses of all 11 nodes


                                        if(Sn[0] == So[0] && Sn[1] == So[1] && Sn[2] ==
So[2] && Sn[3] == So[3] && Sn[4] == So[4] && Sn[5] == So[5] && Sn[6] == So[6] && Sn[7] == So[7] &&
Sn[8] == So[8] && Sn[9] == So[9] && Sn[10] == So[10])
                                                {
                                                        if(Sn[0] == PF1[0] && Sn[1] ==
PF1[1] && Sn[2] == PF1[2] && Sn[3] == PF1[3] && Sn[4] == PF1[4] && Sn[5] == PF1[5] && Sn[6] ==
PF1[6] && Sn[7] == PF1[7] && Sn[8] == PF1[8] && Sn[9] == PF1[9] && Sn[10] == PF1[10])
                                                                Basinsize[0]++;
                                                        else if(Sn[0] == PF2[0] && Sn[1]
== PF2[1] && Sn[2] == PF2[2] && Sn[3] == PF2[3] && Sn[4] == PF2[4] && Sn[5] == PF2[5] && Sn[6] ==
PF2[6] && Sn[7] == PF2[7] && Sn[8] == PF2[8] && Sn[9] == PF2[9] && Sn[10] == PF2[10])
                                                                Basinsize[1]++;
                                                        else if(Sn[0] == PF3[0] && Sn[1]
== PF3[1] && Sn[2] == PF3[2] && Sn[3] == PF3[3] && Sn[4] == PF3[4] && Sn[5] == PF3[5] && Sn[6] ==
PF3[6] && Sn[7] == PF3[7] && Sn[8] == PF3[8] && Sn[9] == PF3[9] && Sn[10] == PF3[10])
                                                                Basinsize[2]++;
                                                        else if(Sn[0] == PF4[0] && Sn[1]
== PF4[1] && Sn[2] == PF4[2] && Sn[3] == PF4[3] && Sn[4] == PF4[4] && Sn[5] == PF4[5] && Sn[6] ==
PF4[6] && Sn[7] == PF4[7] && Sn[8] == PF4[8] && Sn[9] == PF4[9] && Sn[10] == PF4[10])
                                                                Basinsize[3]++;
                                                        else if(Sn[0] == PF5[0] && Sn[1]
== PF5[1] && Sn[2] == PF5[2] && Sn[3] == PF5[3] && Sn[4] == PF5[4] && Sn[5] == PF5[5] && Sn[6] ==
PF5[6] && Sn[7] == PF5[7] && Sn[8] == PF5[8] && Sn[9] == PF5[9] && Sn[10] == PF5[10])
                                                                Basinsize[4]++;
                                                        else if(Sn[0] == PF6[0] && Sn[1]
== PF6[1] && Sn[2] == PF6[2] && Sn[3] == PF6[3] && Sn[4] == PF6[4] && Sn[5] == PF6[5] && Sn[6] ==
PF6[6] && Sn[7] == PF6[7] && Sn[8] == PF6[8] && Sn[9] == PF6[9] && Sn[10] == PF6[10])
                                                                Basinsize[5]++;
                                                        else if(Sn[0] == PF7[0] && Sn[1]
== PF7[1] && Sn[2] == PF7[2] && Sn[3] == PF7[3] && Sn[4] == PF7[4] && Sn[5] == PF7[5] && Sn[6] ==
PF7[6] && Sn[7] == PF7[7] && Sn[8] == PF7[8] && Sn[9] == PF7[9] && Sn[10] == PF7[10])
                                                                Basinsize[6]++;
                                                }

                                        }//end of while loop for the fixed point

                                }

        printf("Cln3\tMBF\tSBF\tCln1,2\tCdh1\tSwi5\tCdc20&Cdc14\tClb5,6\tSic1\tClb1,2\tMcm1&SFF\t
Basinsize \n");

        printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t\t%d\t%d\t%d\t%d\t\t%d\n", PF1[0], PF1[1], PF1[2],
PF1[3], PF1[4], PF1[5], PF1[6], PF1[7], PF1[8], PF1[9], PF1[10], Basinsize[0]);
```

```
		printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t\t%d\t%d\t%d\t%d\t\t%d\n", PF2[0], PF2[1], PF2[2],
PF2[3], PF2[4], PF2[5], PF2[6], PF2[7], PF2[8], PF2[9], PF2[10], Basinsize[1]);

		printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t\t%d\t%d\t%d\t%d\t\t%d\n", PF3[0], PF3[1], PF3[2],
PF3[3], PF3[4], PF3[5], PF3[6], PF3[7], PF3[8], PF3[9], PF3[10], Basinsize[2]);

		printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t\t%d\t%d\t%d\t%d\t\t%d\n", PF4[0], PF4[1], PF4[2],
PF4[3], PF4[4], PF4[5], PF4[6], PF4[7], PF4[8], PF4[9], PF4[10], Basinsize[3]);

		printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t\t%d\t%d\t%d\t%d\t\t%d\n", PF5[0], PF5[1], PF5[2],
PF5[3], PF5[4], PF5[5], PF5[6], PF5[7], PF5[8], PF5[9], PF5[10], Basinsize[4]);

		printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t\t%d\t%d\t%d\t%d\t\t%d\n", PF6[0], PF6[1], PF6[2],
PF6[3], PF6[4], PF6[5], PF6[6], PF6[7], PF6[8], PF6[9], PF6[10], Basinsize[5]);

		printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t\t%d\t%d\t%d\t%d\t\t%d\n", PF7[0], PF7[1], PF7[2],
PF7[3], PF7[4], PF7[5], PF7[6], PF7[7], PF7[8], PF7[9], PF7[10], Basinsize[6]);

					}//case for default dynamic rule



					else if (option3 == 2)
					{
						printf("Please assign values for Ag, Ar and td(Note: The
value of Ar must larger or equal to that of Ag). \n");
						scanf("%d%d%d", &Ag, &Ar, &td);

						printf("Ag = %d\t Ar = %d\t td = %d\n", Ag, Ar, td);

						for(i=0;i<=10;i++)
						{
							Sum[i] = 0;
						}

						for(i=0;i<=4;i++)
						{
							Selfdeg[i] = 1;
						}

						for(count=1;count<=2047;count++)
	{
		for(j=0;j<=10;j++)
		{
		if((count%(int)pow(2,j+1))<(int)pow(2,j))
			Sn[j] = 0;
		else if((count%(int)pow(2,j+1))>=(int)pow(2,j))
			Sn[j] = 1;
		}

		for(i=0;i<=4;i++)
		{
			Selfdeg[i] = 1;
		}

		timeinterval = 0;

	while(timeinterval != td)
						{
						for(i=0;i<=10;i++)
						{
							Sum[i] = 0;
						}

						if (firstentry == 1)
						{
```

```
		for(i=0;i<=10;i++)
		{
			Sn[i] = So[i];
		}
		firstentry = 0;
}

if (Sn[0] == 1)//self-degradation
{
		Sum[1] = Sum[1] + Ag;
		Sum[2] = Sum[2] + Ag;
		Sum[0] = -1;
}

if (Sn[1] == 1)
{
		Sum[7] = Sum[7] + Ag;
}

if (Sn[2] == 1)
{
		Sum[3] = Sum[3] + Ag;
}

if (Sn[3] == 1)//self=degradation
{
		if (Sn[2] == 0)
		{
			Sum[4] = Sum[4] - Ar;
			Sum[8] = Sum[8] - Ar;
			Sum[3] = -1;
		}

		else if(Sn[2] == 1)
		{
			Sum[4] = Sum[4] - Ar;
			Sum[8] = Sum[8] - Ar;
		}
}

if (Sn[4] == 1)
{
		Sum[9] = Sum[9] - Ar;
}

if (Sn[5] == 1)//self-degradation
{
		if (Sn[6] + Sn[10] - Sn[9] == 0)
		{
			Sum[8] = Sum[8] + Ag;
			Sum[5] = -1;
		}

		else if(Sn[6] + Sn[10] - Sn[9] > 0)
		{
			Sum[8] = Sum[8] + Ag;
		}
}

if (Sn[6] == 1)//self-degradation
{
		if (Sn[9] + Sn[10] == 0)
		{
			Sum[4] = Sum[4] + Ag;
			Sum[5] = Sum[5] + Ag;
			Sum[8] = Sum[8] + Ag;
			Sum[7] = Sum[7] - Ar;
```

```
                Sum[9] = Sum[9] - Ar;
                Sum[6] = -1;
        }

        else if (Sn[9] + Sn[10] > 0)
        {
                Sum[4] = Sum[4] + Ag;
                Sum[5] = Sum[5] + Ag;
                Sum[8] = Sum[8] + Ag;
                Sum[7] = Sum[7] - Ar;
                Sum[9] = Sum[9] - Ar;
        }
}

if (Sn[7] == 1)
{
        Sum[9] = Sum[9] + Ag;
        Sum[10] = Sum[10] + Ag;
        Sum[4] = Sum[4] - Ar;
        Sum[8] = Sum[8] - Ar;
}

if (Sn[8] == 1)
{
        Sum[7] = Sum[7] - Ar;
        Sum[9] = Sum[9] - Ar;
}

if (Sn[9] == 1)
{
        Sum[6] = Sum[6] + Ag;
        Sum[10] = Sum[10] + Ag;
        Sum[1] = Sum[1] - Ar;
        Sum[2] = Sum[2] - Ar;
        Sum[4] = Sum[4] - Ar;
        Sum[5] = Sum[5] - Ar;
        Sum[8] = Sum[8] - Ar;
}

if (Sn[10] == 1)//self-degradation
{
        if (Sn[9] == 0)
        {
                Sum[5] = Sum[5] + Ag;
                Sum[6] = Sum[6] + Ag;
                Sum[9] = Sum[9] + Ag;
                Sum[10] = -1;
        }

        else if(Sn[9] == 1)
        {
                Sum[5] = Sum[5] + Ag;
                Sum[6] = Sum[6] + Ag;
                Sum[9] = Sum[9] + Ag;
        }
}
//update the value for status conditions for all nodes

for(i=0;i<=10;i++)
        {
                Temp[i] = Sn[i];
        }//Store the old statuses in a Temp array

for(i=0;i<=10;i++)
        {
                So[i] = Temp[i];
        }//update the old statuses for 11 nodes
```

```
for(i=0;i<=10;i++)
{
	if (Sum[i] > 0)
		Sn[i] = 1;
	else if (Sum[i] < 0)
		Sn[i] = 0;
	else Sn[i] = So[i];
}//The new statuses of all 11 nodes

if(Sn[0] == So[0] && Sn[1] == So[1] && Sn[2] == So[2] &&
Sn[3] == So[3] && Sn[4] == So[4] && Sn[5] == So[5] && Sn[6] == So[6] && Sn[7] == So[7] && Sn[8] ==
So[8] && Sn[9] == So[9] && Sn[10] == So[10])
{
	timeinterval++;
}

else timeinterval = 0;

}//end of while loop for the fixed point


if(Sn[0] == So[0] && Sn[1] == So[1] && Sn[2] == So[2] &&
Sn[3] == So[3] && Sn[4] == So[4] && Sn[5] == So[5] && Sn[6] == So[6] && Sn[7] == So[7] && Sn[8] ==
So[8] && Sn[9] == So[9] && Sn[10] == So[10])
	{
		if(Sn[0] == PF1[0] && Sn[1] == PF1[1]
&& Sn[2] == PF1[2] && Sn[3] == PF1[3] && Sn[4] == PF1[4] && Sn[5] == PF1[5] && Sn[6] == PF1[6] &&
Sn[7] == PF1[7] && Sn[8] == PF1[8] && Sn[9] == PF1[9] && Sn[10] == PF1[10])
			Basinsize[0]++;
		else if(Sn[0] == PF2[0] && Sn[1] ==
PF2[1] && Sn[2] == PF2[2] && Sn[3] == PF2[3] && Sn[4] == PF2[4] && Sn[5] == PF2[5] && Sn[6] ==
PF2[6] && Sn[7] == PF2[7] && Sn[8] == PF2[8] && Sn[9] == PF2[9] && Sn[10] == PF2[10])
			Basinsize[1]++;
		else if(Sn[0] == PF3[0] && Sn[1] ==
PF3[1] && Sn[2] == PF3[2] && Sn[3] == PF3[3] && Sn[4] == PF3[4] && Sn[5] == PF3[5] && Sn[6] ==
PF3[6] && Sn[7] == PF3[7] && Sn[8] == PF3[8] && Sn[9] == PF3[9] && Sn[10] == PF3[10])
			Basinsize[2]++;
		else if(Sn[0] == PF4[0] && Sn[1] ==
PF4[1] && Sn[2] == PF4[2] && Sn[3] == PF4[3] && Sn[4] == PF4[4] && Sn[5] == PF4[5] && Sn[6] ==
PF4[6] && Sn[7] == PF4[7] && Sn[8] == PF4[8] && Sn[9] == PF4[9] && Sn[10] == PF4[10])
			Basinsize[3]++;
		else if(Sn[0] == PF5[0] && Sn[1] ==
PF5[1] && Sn[2] == PF5[2] && Sn[3] == PF5[3] && Sn[4] == PF5[4] && Sn[5] == PF5[5] && Sn[6] ==
PF5[6] && Sn[7] == PF5[7] && Sn[8] == PF5[8] && Sn[9] == PF5[9] && Sn[10] == PF5[10])
			Basinsize[4]++;
		else if(Sn[0] == PF6[0] && Sn[1] ==
PF6[1] && Sn[2] == PF6[2] && Sn[3] == PF6[3] && Sn[4] == PF6[4] && Sn[5] == PF6[5] && Sn[6] ==
PF6[6] && Sn[7] == PF6[7] && Sn[8] == PF6[8] && Sn[9] == PF6[9] && Sn[10] == PF6[10])
			Basinsize[5]++;
		else if(Sn[0] == PF7[0] && Sn[1] ==
PF7[1] && Sn[2] == PF7[2] && Sn[3] == PF7[3] && Sn[4] == PF7[4] && Sn[5] == PF7[5] && Sn[6] ==
PF7[6] && Sn[7] == PF7[7] && Sn[8] == PF7[8] && Sn[9] == PF7[9] && Sn[10] == PF7[10])
			Basinsize[6]++;
	}



}

printf("Cln3\tMBF\tSBF\tCln1,2\tCdh1\tSwi5\tCdc20&Cdc14\tClb5,6\tSic1\tClb1,2\tMcm1&SFF\t
Basinsize \n");
	printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t\t%d\t%d\t%d\t%d\t\t%d\n", PF1[0], PF1[1],
PF1[2], PF1[3], PF1[4], PF1[5], PF1[6], PF1[7], PF1[8], PF1[9], PF1[10], Basinsize[0]);
	printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t\t%d\t%d\t%d\t%d\t\t%d\n", PF2[0], PF2[1],
PF2[2], PF2[3], PF2[4], PF2[5], PF2[6], PF2[7], PF2[8], PF2[9], PF2[10], Basinsize[1]);
```

```
                printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t\t%d\t%d\t%d\t%d\t\t%d\n", PF3[0], PF3[1],
PF3[2], PF3[3], PF3[4], PF3[5], PF3[6], PF3[7], PF3[8], PF3[9], PF3[10], Basinsize[2]);
                printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t\t%d\t%d\t%d\t%d\t\t%d\n", PF4[0], PF4[1],
PF4[2], PF4[3], PF4[4], PF4[5], PF4[6], PF4[7], PF4[8], PF4[9], PF4[10], Basinsize[3]);
                printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t\t%d\t%d\t%d\t%d\t\t%d\n", PF5[0], PF5[1],
PF5[2], PF5[3], PF5[4], PF5[5], PF5[6], PF5[7], PF5[8], PF5[9], PF5[10], Basinsize[4]);
                printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t\t%d\t%d\t%d\t%d\t\t%d\n", PF6[0], PF6[1],
PF6[2], PF6[3], PF6[4], PF6[5], PF6[6], PF6[7], PF6[8], PF6[9], PF6[10], Basinsize[5]);
                printf("%d\t%d\t%d\t%d\t%d\t%d\t%d\t\t%d\t%d\t%d\t%d\t\t%d\n", PF7[0], PF7[1],
PF7[2], PF7[3], PF7[4], PF7[5], PF7[6], PF7[7], PF7[8], PF7[9], PF7[10], Basinsize[6]);


                                }//cases for other dynamic rules

                                else printf("Wrong Number!!\n");

                                printf("\nBack to main list, please enter 0; to continue, please press
1: ");
                                scanf("%d", &option2);
                        }//end of the while loop for option2


                }

                else
                {
                        printf("\nNo such option!\n");
                        printf("To investigate the trajectories in state space; please enter 1. \n");
                        printf("To do statistics of the big fixed points of the cell-cycle network; please
enter 2. \n");
                        printf("To quit the program; please enter 3. \n");
                        printf("Please select one option: ");
                        scanf("%d", &option);
                }

                printf("\nTo investigate the trajectories in state space; please enter 1. \n");
                printf("To do statistics of the big fixed points of the cell-cycle network; please enter 2.
\n");
                printf("To quit the program; please enter 3. \n");
                printf("Please select one option: ");
                scanf("%d", &option);

        }//the end of the biggest while loop

}
```