CLARKSON UNIVERSITY

Networked Networks: Uncovering Hierarchical Scales of Spatial-Temporal Dynamics

A dissertation

by

Jie Sun

Department of Mathematics and Computer Science

Submitted in partial fullfillment of the requirements

for the degree of

Doctor of Philosophy

Applied Mathematics

Date

Accepted by the Graduate School

Date	Dean

CLARKSON UNIVERSITY

The undersigned have examined the dissertation entitled **Networked Networks: Uncovering Hierarchical Scales of Spatial-Temporal Dynamics** presented by **Jie Sun** a candidate of the degree of **Doctor of Philosophy** and hereby certify that it is worthy of acceptance.

Date	Erik Bollt (Advisor)

Examining Committee

Erik Bollt	**Date**
Daniel ben-Avraham	**Date**
Takashi Nishikawa	**Date**
Joseph Skufca	**Date**
Aaron Luttman	**Date**

Networked Networks: Uncovering Hierarchical Scales of Spatial-Temporal Dynamics

by

Jie Sun

Abstract

Networked Networks: Uncovering Hierarchical Scales of Spatial-Temporal Dynamics

by

Jie Sun

Doctor of Philosophy in Applied Mathematics

Clarkson University

The first, and main part of the thesis is the exploration of coupled oscillator networks (OSN). OSN are often used as models for complicated systems that consist of interacting components. It is desirable to have reduced order models for OSNs. However, the type of averaging which leads to a reduced order model has not been well dened in the eld of dynamical systems. In this thesis we develop analysis on how to reduce order modeling an OSN. For an OSN, usual clustering methods for graphs do not necessarily define meaningful dynamical relevant groups. We show that the interplay between dynamical and structural heterogeneity is key to uncovering the spatial scales. Motivated by the idea of shadowing, we develop a novel way to assess the quality of a reduced order model of an OSN especially for chaotic oscillators.

The second part of this thesis is about a computational approach for efficiently computing important network statistics of an evolving network. Instead of starting from scratch, we develop updating schema updates important network statistics upon structural changes to the network. Both local statistics such as degree and global statistics such as path lengths can be updated efficiently, allowing us to analyze the actual evolution of network statistics, explore new properties of time dependent networks and search for common features of different types of such networks.

The third part of the thesis emphasized on various problems in network modeling, including the problem of how to compress sparse graphs, a new class of network model called sequence networks, and a new notion of connectedness called greedy connectedness (or connectivity) for networks that are embedded in metric spaces.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to first thank my family, especially my parents Wenbin Sun (father) and Yueqin Zhu (mother) for their love and support, without which I would have hardly achieved anything. My grandparents have also been an important source of motivation for my study and research. Special thank to my wife Xi Chen for her accompany, and her parents for being supportive.

During my graduate study and reseach, my advisor Professor Erik Bollt has been giving me tremendous help in my study and research. He has shown me not only how to become a mathematician and scientist, but also how to become a great person. I am grateful to all his patience, encouragement, and support. Professor Daniel ben-Avraham has co-advised me in a few research projects through which I not only get to learn from his wisdom but also his view of science and life. I want to thank him for spending time shaping me into a real scientist like himself, and his green teas in reminding me of my country (China). I also want to thank Professor Takashi Nishikawa for co-advising me in the research projects about dynamics on networks and sharing with me his research experience. Professor Joseph Skufca also has given me a lot of advice in research. I appreciate all the useful discussion with him.

I am indebted to two of my collaborators, Dr. Attilio Milanese and Dr. James Bagrow, both of which are former Clarkson students. During a project we worked on together, I learned a lot from Dr. Milanese, especially his expertise in numerical simulation and the art of computer programming. Dr. Bagrow has also been helpful in my research especially in complex networks.

I wish to thank Michael Felland and Professor Scott Fulton for educating me in some fundamentals of advanced mathematics which have impacted my research. I also wish to thank my current lab mates Jiongxuan Zheng, Ranil Basnayake, Sean Kramer, and Dr. Rana Parshad; and former students Dr. Hernan Rozenfeld, Dr. Naratip Santitissadeekorn, and Dr. Chen Yao. In particular, Dr. Chen Yao gave me a lot of help when I first came to the USA.

Warmest thanks are also due to the Department of Mathematics and Computer Science of Clarkson University, which supported me as a teaching assistant during my first year in graduate school; and Army Research Office (51950-MA) from which

I receive a research assistantship since my second year, working on a project under the supervision of Professor Erik Bollt.

A special thank to Professor Mason Porter who I visited with Professor Erik Bollt in the September and October of 2009. I like to thank the Mathematical Institute of the University of Oxford for their hospitality and Army Research Office (51950-MA) for financial support of this trip.

Finally, let me thank Professors Erik Bollt, Daniel ben-Avraham, Takashi Nishikawa, Joseph Skufca, and Aaron Luttman for serving as committee members of my thesis; and Sean Kramer and Claudette Foisy for their comments on my thesis draft. Of course, I will be the only one who is responsible for all the typo and mistakes appearing in this thesis.

Chapter 1

Introduction

1.1 Dynamics on Networks: Multi-scale Modeling of Spatial-Temporal Systems

Coupled oscillator networks (OSN) are often used as models for complicated dynamical systems that consist of interacting components which evolve in time. Although it is usually assumed that the oscillators are identical, an OSN may in general (and in practice) contain oscillators that are not identical. Furthermore, those non identical oscillators may connect through some highly nontrivial topology. It is desirable and useful to have reduced order models for OSNs.

However, the type of averaging which leads to a reduced order model has not been well defined in the field of dynamical systems. Specifically, how to judge how good an 'average model' is in the case of an OSN consisting of non identical oscillators is not straightforward. Direct comparison of some average time series generated by an OSN and time series coming from its average model might be misleading specifically in the case of chaotic systems.

Motivated by the idea of shadowing, we have developed a novel way to assess the quality of a reduced order model of an OSN. Based on our approach, a model's quality can be naturally measured for either modeling an OSN by a single oscillator in the

case of complete synchronization or nearly synchronization, or by another simplified OSN with fewer oscillators, which eventually lead to hierarchical scales of the OSN in a dynamical sense. Numerical examples including coupled quadratic maps, coupled Henon oscillators, and coupled Kuramoto oscillators will be shown to illustrate our approach.

Those will be discussed in Chapters 2, 3, and 4.

1.2 Dynamics of Networks: A Computational Approach for Analyzing Time Dependent Networks

Large complex networks have been studied extensively in the past ten years as one important tool for the study of complex systems. Many useful statistics (degree, clustering, path length, spectrum, and so on) have been used to analyze networks surrounding us: social, biological, and technical networks. Efficient computation is key to allowing such analysis for a large network. However, when a network is slowly evolving in time, computing even cheap statistics might be costly, since the time variable could be large. It is appealing to fill this gap and have a tool to analyze the dynamics *of* a network.

In this thesis we introduce a new computational approach called updating schema for efficiently computing important network statistics of an evolving network. Instead of starting from scratch, the updating schema updates important network statistics upon structural changes to the network. Both local statistics such as degree and global statistics such as path lengths can be updated efficiently, allowing us to analyze the actual evolution of network statistics, explore new properties of time dependent networks and search for common features of different types of such networks.

Those will be discussed in Chapters 6 and 7.

1.3 Network Modeling: Graph Compression, Sequence Nets, and Greedy Connectivity

1.3.1 Graph Compression

Given a large network representing some structural information of a real system, how do we efficiently represent such a network? Although matrices (or lists) can be conveniently used, it is unclear whether there exists a better, or optimal, data structure specifically for representing graphical structures. This problem is discussed in Chapter 8, where a heuristic approach is proposed, based on reducing the information storage by exploiting symmetry.

1.3.2 Sequence Nets

A threshold graph is a type of graph where nodes are assigned hidden weights and edges are present only between nodes whose sum of weights exceed a prescribed threshold. Interestingly, threshold graph can be represented equivalently by a sequence of numbers representing nodes, and a deterministic rule on how to connect the nodes. We propose and study a new class of networks called sequence networks based on the observation that a sequence plus a rule can be used to generate graphs. Sequence networks (to be discussed in Chapter 9) have many attractive features such as modular structures which allow high compression, easily computable structural measures—including the possibility of design—and a high degree of compressibility.

1.3.3 Greedy Connectivity

Is a connected graph indeed good for communication or navigation? A maze is an example of a connected graph which is designed to be challenging to navigate; on the other hand, a map of a city provides sufficient information for commutation, even if the density of roads might be lower than that of a maze. The key difference comes from the fact that geographical coordinates of places in a city are usually useful, while those

of a maze are usually of little use. To describe such graphs and their connectedness in a more abstract/rigorous sense, we introduce a concept called greedy connectedness for geographical graphs and study some of its amusing properties in Chapter 10.

1.4 Multiscale Graphical Illustration of the Contents

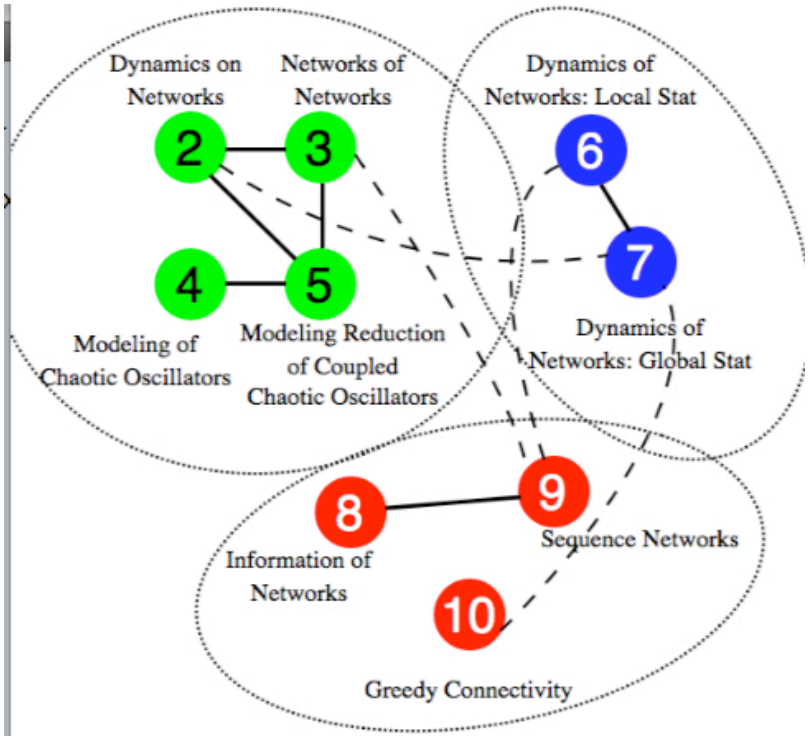

Figure 1.1. **A graphical representation of the interconnections between chapters.** -

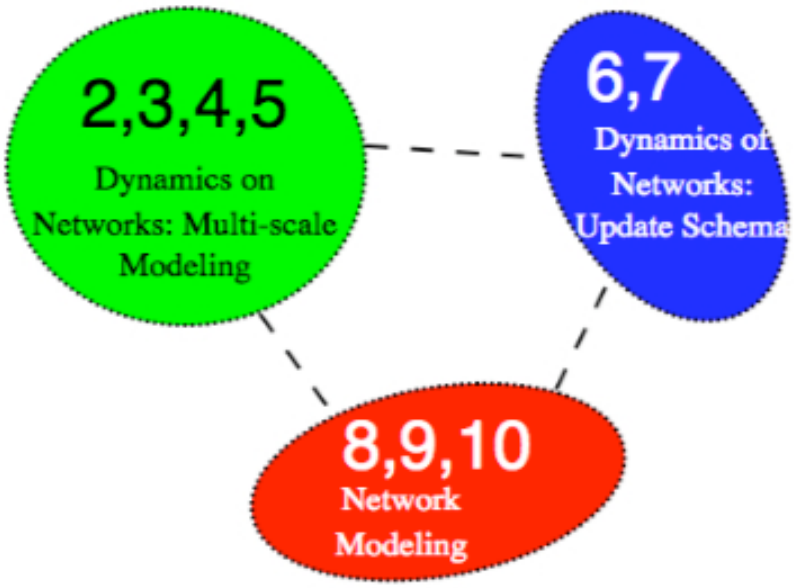

Figure 1.2. **A 'reduced order' representation of the figure shown in Fig. 1.1.**
-

Chapter 2

Dynamics on Networks: Coupled Oscillators, Synchronization

Chaos is known to cause systems to be long-term unpredictable. One consequence is that for any chaotic system, any two typical trajectories which differ even slightly at certain time will eventually diverge from each other. Following this rationale, it seems reasonable to conclude that in reality any two chaotic oscillators will typically generate different trajectories that are uncorrelated, unless they start with the *exact* same initial condition at the *exact* same time, and external perturbations affect the two systems in *exact* the same ways. Indeed, the fact that even chaotic systems can synchronize was a surprise. In a paper by Pecora and Carroll in 1990 (PC90), the authors showed that a key factor in achieving synchronization between chaotic oscillators is effective coupling. For example, take two Lorenz oscillators and add an appropriate linear diffusive coupling between them, then we observe that their difference converges to zero, often exponentially as t goes to infinity. See Fig. 2.1 and Fig. 2.2 for illustrations.

As simple (and maybe artificial) as it seems, the phenomena of synchronization is observed in such a wide range of areas in science and engineering, including laster, electronic circuits, robotic motions, and a few more (SS93; PRK01; BKO$^+$02); such

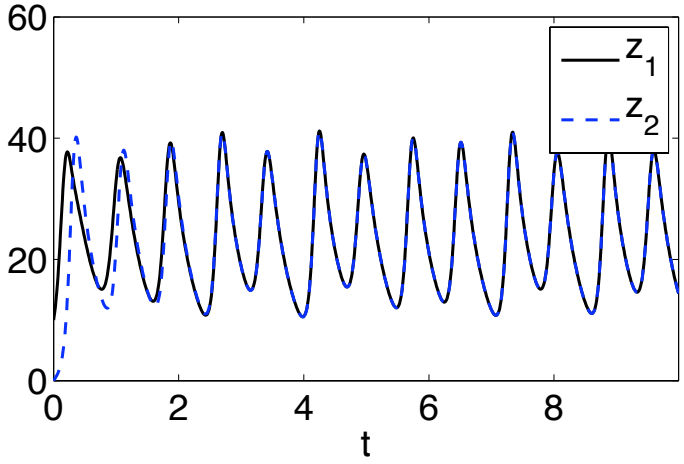

Figure 2.1. **Synchronization of two coupled Lorenz oscillators: z components.** - Shown in the figure are the time series of z components of two coupled Lorenz oscillators. The motion of oscillator 1 is governed by the Lorenz equations described by: $[\dot{x_1}, \dot{y_1}, \dot{z_1}] = [10(-x_1 + y_1), x_1(28 - z_1) - y_1, x_1 y_1 - \frac{8}{3} z_1]$, plus an external force from oscillator 2: $[0.15x_2, 0.15y_2, 0.15z_2]$; and similarly for oscillator 2. In this example the two oscillators start with initial conditions: $[10, 10, 10]$ and $[0, 0, 0]$ respectively; yet they manage to synchronize.

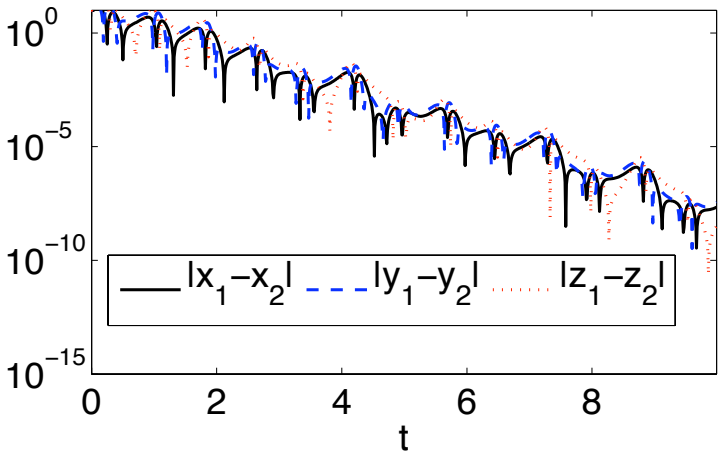

Figure 2.2. **Synchronization of two coupled Lorenz oscillators: convergence of states.** - See Fig. 2.1 for description. Here we show that the x, y, z components of the two oscillators indeed convergences exponentially in this case.

beautiful synchronized patterns are also seen in nature and social life. See Fig. 2.3 and Fig. 2.4 for examples.

Figure 2.3. **Brandt geese flock over Baywood.** - The geese form a beautiful pattern and yet fly highly coherently toward their destination. Picture adopted from: http://baywoodnavy.org/Perlstein/slides/
Brandt%20geese%20flock%20over%20Baywood.jpg

In all the above examples, the different individuals (a chaotic oscillator, a bird, or a human being) certainly cannot guarantee that they always follow *exactly* the motion of the others. Nevertheless, the systems seem to be able to regulate themselves whenever small perturbations occur, bringing the individuals back into synchrony. This feature is nontrivial because of the fact that there is no global coordinator (at least for the second and third examples). In many typical situations, the individuals are only able to communicate *locally* to a finite number of neighbors, and yet achieves excellent synchronization.

In this chapter we study synchronization in terms of coupled oscillator network, a system of differential (or difference) equations coupled through some underlying network structure (which encodes the possible communications between individuals). In section 1 we review the definition of a coupled oscillator network; in section 2 we review the well-known master stability function approach, a powerful tool in analyzing the stability of complete synchronization; in section 3 we develop generalized version of the master stability functions to allow the stability analysis go beyond identical case; in section 4 a famous example of phase oscillators, namely the Kuramoto oscil-

Figure 2.4. **TaiChi** - Tai Chi, a traditional Chinese martial art, was performed during the Opening Ceremony for the 2008 Beijing Summer Olympics at the National Stadium on August 8, 2008 in Beijing, China. Despite the large number of martial artists participated as shown, their movements (at least seen from the picture) are surprisingly consistent. Picture adopted from: http://en.beijing2008.cn/ceremonies/photos/openingceremony/performances/n214517049.shtml

lators, is studied, with emphasis on designing, for a given set of mismatched phase oscillators, an optimal placement for the purpose of achieving optimal synchrony.

2.1 Coupled Oscillator Network

2.1.1 Equations of Motion

To capture the property of a complex system that consists of interacting individuals who themselves move in time, a typical mathematical model need to take into account the dynamics of individuals, and how they interact. To this end, the following equations are proposed, similar to (PC98), but allows the individual dynamics to be different:

$$\dot{w}_i = f(w_i, \mu_i) - \sigma \sum_{j=1}^{n} l_{ij} h(w_j), \quad i = 1, 2, \ldots, n, \tag{2.1}$$

where $f : \Re^{m\times p} \to \Re^m$ is the parameterized dynamics of an isolated unit; $w_i \in \Re^m$ is the dynamical variable for the ith unit; $\mu_i \in \Re^p$ is the corresponding parameter; $L \in \Re^{n\times n}$ is the graph Laplacian (unnormalized) [1]; $h : \Re^m \to \Re^m$ is a uniform coupling function; and $\sigma \in \Re$ is the uniform coupling strength (usually > 0 for diffusive coupling).

Note that we can represent the whole system conveniently by using Kronecker product representation:

$$\dot{\boldsymbol{w}} = \boldsymbol{f}(\boldsymbol{w}, \boldsymbol{\mu}) - g \cdot L \otimes H(\boldsymbol{w}), \tag{2.2}$$

where $\boldsymbol{w}= [w_1^T, w_2^T, ...w_N^T]^T$ is a column vector of all the dynamic variables, and likewise for $\boldsymbol{\mu}$ and $\boldsymbol{f}$; and $\otimes$ is the usual Kronecker product (or direct product) (Ber92).

The question of interest is that, under what conditions can the highly complicated coupled oscillator system exhibit coherent behavior, i.e., the oscillators *completely synchronize*, or more precisely,

$$\lim_{t\to\infty} ||w_i - w_j|| = 0, \forall i, j. \tag{2.3}$$

2.1.2 Graph Laplacian

For a given undirected graph with associated adjacency matrix $A = [a_{ij}]_{n\times n}$, where a_{ij} is the weight on the edge between nodes i and j, the *graph Laplacian* L is a square matrix of the same size of A, whose entries are defined as: $l_{ij} = -a_{ij}$ if $i \neq j$; and $l_{ii} = \sum_j a_{ij}$. Note that the assumption that L being symmetric is the same as assuming A being symmetric, i.e., the graph being undirected. In this case, L is semi positive definite, and has diagnalization form:

$$L = P\Lambda P^T, \tag{2.4}$$

where

$$\Lambda = diag([\lambda_1, \lambda_2, ..., \lambda_n]) \tag{2.5}$$

[1] We only deal with graph Laplacians that are constant, and symmetric (and thus semi-positive definite) for the reason of clarity. Treatment of general constant graph Laplacians can be found by techniques proposed in (NM06b; NM06a); when the graph Laplacian is *time dependent*, see (SBR06) and the references therein

is a diagonal matrix whose diagonal elements are the eigenvalues of L, ordered in a nondecreasing way, so that $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$; P is an orthonormal matrix whose columns are the corresponding eigenvectors.

One can check that by definition, L always has 0 as one of its eigenvalues, with the corresponding eigenvector $[1, ..., 1]^T$. Indeed 0 has to be the smallest eigenvalue of L, since L is semi positive definite. Furthermore, λ_2 (which is also known as the *algebraic connectivity* of a graph (Fie89)) determines whether the graph is *connected* (i.e., there is a path between any pair of nodes) or not. The graph is connected if and only if $\lambda_2 > 0$. We will, in the following, assume that this is always the case, unless specified.

2.2 Complete Synchronization: Master Stability Function Analysis

2.2.1 Variational Equations

In the simplest case where $\mu_1 = \mu_2 = ... = \mu_n$, a now famous approach named master stability function (MSF) analysis (PC98) was introduced by Pecora and Carroll in 1998, which allows one to answer the question of how synchronization stability depends on the dynamics, coupling form, and network topology. In the following we review this MSF approach.

It is convenient to rewrite equation in this case, where now we have:

$$\dot{w_i} = f(w_i) - \sigma \sum_{j=1} l_{ij} h(w_j), \quad i = 1, 2, \ldots, n, \tag{2.6}$$

Note that in this case, if one plugs in, at any time $t^\star$, $w_1 = w_2 = ... = w_n$ in the above equation, the second term will become zero, for each i. Since the system is autonomous, we have, $w_1(t) = w_2(t) = ... = w_n(t)$, for any $t > t^\star$. In particular, if we do so when $t^\star = 0$, i.e., start with the situation that all the oscillators are of the same state, then they continue to be so.

Such ideal situation rarely happens in any physical settings, due to various reasons (perturbation of various forms). Suppose the system is perturbed from its ideal synchrony, so that for oscillator i, we have:

$$w_i = s + \eta_i \tag{2.7}$$

where s represents a collective variable, and η is the perturbation term. Assuming that $\eta_i \ll 1$, linearization around the synchronous state s, we have, for oscillator i,

$$\begin{aligned} \dot{\eta_i} &= Df(s)\eta_i - \sigma \sum_j \left[l_{ij} h(s) + Dh(s)\eta_j \right] \\ &= Df(s)\eta_i - \sigma Dh(s) \sum_j l_{ij}\eta_j \end{aligned} \tag{2.8}$$

Putting the variational equations together, and omit the argument s for convenience, we have:

$$\dot{\boldsymbol{\eta}} = \Big[I_n \otimes Df - \sigma \cdot L \otimes Dh \Big] \boldsymbol{\eta}, \tag{2.9}$$

i.e., a homogeneous linear system. Making use of the fact that $L = P\Lambda P^T$, usual change of coordinates $\boldsymbol{\zeta} = (P^T \otimes I_n)\boldsymbol{\eta}$ leads to:

$$\dot{\boldsymbol{\zeta}} = \Big[I_n \otimes Df - \sigma \cdot \Lambda \otimes Dh \Big], \tag{2.10}$$

the uncoupled set of equations (in the eigen-basis):

$$\dot{\zeta_i} = [Df - \sigma\lambda_i Dh]\zeta_i \tag{2.11}$$

Since $\lambda_1 = 0$, we have, for $i = 1$,

$$\dot{\zeta_1} = [Df]\zeta_1, \tag{2.12}$$

a linearized equation around the state of the dynamics governed by $\dot{s} = f(s)$, indicating the infinitesimal motion tangent to the trajectory s.

Thus, if all the other $\zeta_i \to 0$, then all the transverse perturbation dies out, and a slightly perturbed state will come back to the synchronous state s. This requires that the solution of Eq. (2.11) goes to zero for all $i \geq 2$.

Note that since the input of Df and Dh is the state variable of the synchronous state s, which is governed by $\dot{s} = f(s)$, those homogeneous equations would have

variable coefficients, in which case the 'eigenvalues' of Df and Dh has little to do with the actual behavior of the solution, unlike in the case of constant coefficient linear equations.

Nevertheless, for chaotic orbits that have well-defined Lyapunov exponents, the requirement that $\zeta_i \to 0$ is equivalent as requiring the largest Lyapunov exponent of the flow of Eq. (2.11) associated with the motion $\dot{s} = f(s)$ being negative. Thus, the local synchronization stability can be determined by checking the largest Lyapunov exponent of Eq. (2.11) for all $i \geq 2$. Synchronization is locally stable if and only if for each $i \geq 2$, the corresponding largest Lyapunov exponent is negative.

2.2.2 Master Stability Function: Derivation

Having noticed that for given form of the individual dynamics f and coupling function h, the form of Eq. (2.11) is the same except for the term $\sigma\lambda_i$, Pecora and Carroll in (PC98) introduced a stability function called master stability function Θ as a function of α, where $\Theta(\alpha)$ equals the largest Lyapunov exponent of the flow

$$\dot{\zeta} = [Df(s) - \alpha Dh(s)]\zeta \tag{2.13}$$

associated with $\dot{s} = f(s)$.

This function can be studied for given f and h, regardless of the structure of the network. Once this function is given, for an *arbitrary* coupled oscillator network with f and h being the individual dynamics and coupling function, its synchronization stability can be determined by simply checking whether $\Theta(\sigma\lambda_i) < 0$ for all $i \geq 2$, if it is, then synchronization is locally stable; and not if not.

2.2.3 Master Stability Function: Examples

Let us first look at a simple example, where f is a linear operator, represented by the matrix:

$$\begin{bmatrix} -2 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0.5 \end{bmatrix} \tag{2.14}$$

so that without coupling, two oscillators starting from different initial conditions will typically diverge from each other, with distance proportional to $e^{0.5t}$. Suppose that we have three different coupling functions h_1, h_2, h_3, which are again linear operators, described by:

$$h_1(w) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} w, \tag{2.15}$$

$$h_2(w) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} w, \tag{2.16}$$

and

$$h_3(w) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} w, \tag{2.17}$$

respectively. The master stability equation for f and h_k will have the form:

$$\dot{s} = [Df - \alpha Dh_k]s, \tag{2.18}$$

which is a homogeneous linear equation with constant coefficients, and so the master stability function $\Theta_k(\alpha)$ is simply the largest eigenvalue of the matrix $[Df - \alpha Dh_k]$. This function is computed for all three coupling functions, and shown in Fig. 2.5. Note that for certain coupling functions, increasing the coupling strength might actually destroy the stability of synchronization (case h_2); and there are specific coupling function for which synchronization stability is simply not attainable (case h_3).

Turn to chaotic systems, consider f being governed by the *Rossler equations* (Ros76):

$$\begin{aligned} \dot{x} &= -y + z, \\ \dot{y} &= x + 0.2y, \\ \dot{z} &= 0.2 + (x - 7)z, \end{aligned} \tag{2.19}$$

with coupling function $h([x, y, z]^T) = [x, 0, 0]^T$. The corresponding MSF is computed by numerical technique suggested in (WSSV85) and shown in Fig. 2.6. In Fig. 2.7 we

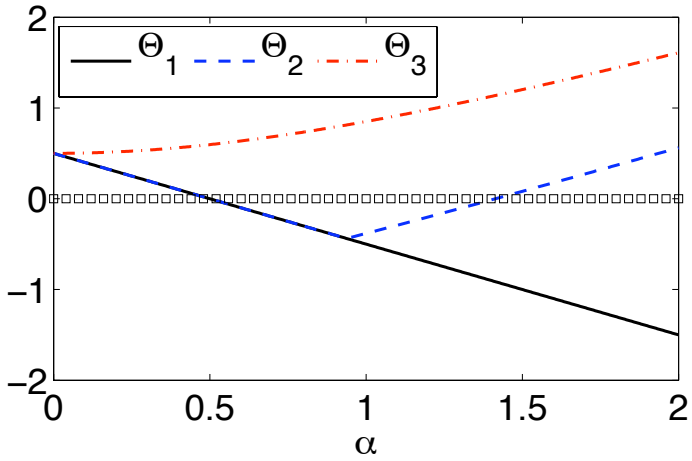

Figure 2.5. **Master stability functions for a linear system.** - Here f is a linear operator represented by the matrix $diag([-2,-1,0.5])$. Shown in the figure are master stability functions for three different coupling functions h_1, h_2, h_3 as described by Eq. (2.15), Eq. (2.16),and Eq. (2.17) respectively. The master stability equation can be derived simply according to Eq. (2.18), and thus the function $\Theta_k(\alpha)$ in this case is simply the largest eigenvalue of the matrix $[Df - \alpha Dh_k]$.

show an example of two coupled Rossler oscillators with different coupling strengths: $\sigma = 0, 1, 2, 3$. Correspondingly, the synchronization stability can be determined by $\Theta(\lambda_i \sigma)$ $(i \geq 2)$ for each system. In this case we have $\lambda_1 = 0, \lambda_2 = 1$. By looking at Fig. 2.6, it is clear that the cases where $\sigma = 1, 2$ (so $\alpha = 2, 4$) correspond to stable synchronization; and the cases where $\sigma = 0, 3$ (so $\alpha = 0, 6$) correspond to unstable synchronization.

2.3 Nearly Synchronization: Generalized Master Stability Functions

2.3.1 Motivation

System (2.1) has been studied mostly in the case in which the parameter μ_i is the same for each individual oscillator, often resulting in *complete synchronization* where

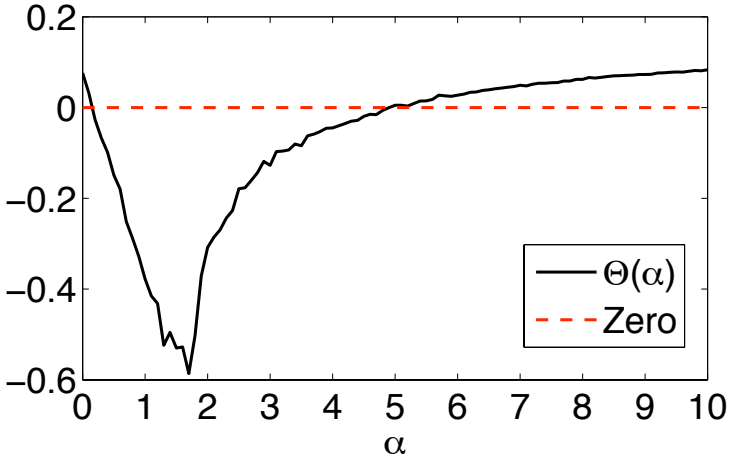

Figure 2.6. **Master stability function for Rossler equations with x coupling.** - This picture shows the master stability function for the Rossler equations (2.19) with coupling function $h([x, y, z]^T) = [x, 0, 0]^T$.

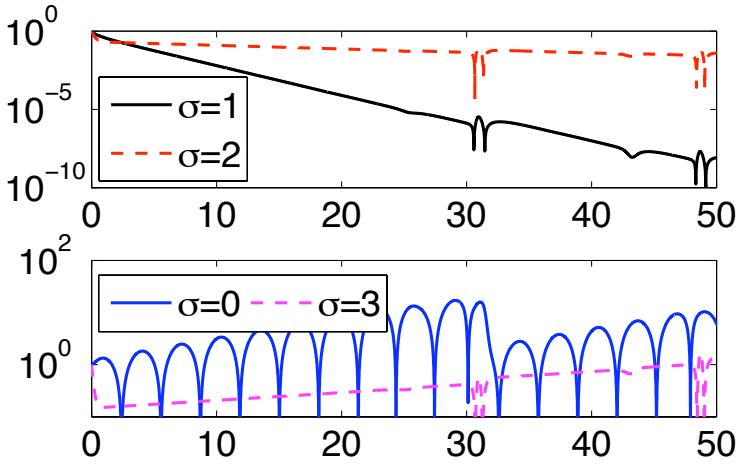

Figure 2.7. **Synchronization of coupled Rossler oscillators.** - We show different dynamic behavior for two coupled Rossler oscillators with x coupling when changing the coupling strength σ. In both panels the horizontal axis is t, while the vertical axis shows $|x_1(t) - x_2(t)|$ in different cases. In the upper panel we show the cases where σ is 1 and 2, which correspond to α being 2 and 4 in Fig. 2.6. In these cases synchronization is stable, and the rate of converge can be estimated by the value of $\Theta(\alpha)$ in Fig. 2.6. On the other hand, when σ is 0 or 3, synchronization becomes unstable, since in those cases α would be 0 and 6, corresponding to positive values of Θ.

$$\max_{i,j} ||w_i(t) - w_j(t)|| \to 0 \text{ as } t \to \infty.$$

The stability of such states can be analyzed by master stability functions (MSF), as illustrated in the previous section.

However, a noiseless system with exactly the same parameters is impossible in practice. What happens if the oscillators are non-identical? What if they are *nearly identical*? Fig. 2.8 serves as an illustration.

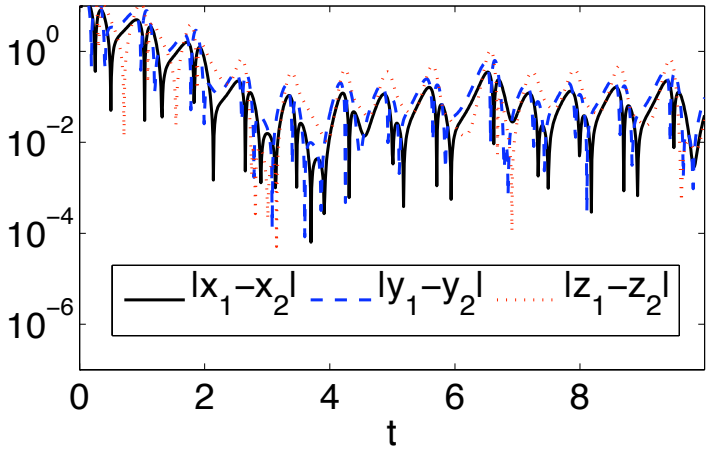

Figure 2.8. **Synchronization error of two coupled mismatched Lorenz oscillators.** - Here we show how synchronization is affected by parameter mismatch. The system consists of two coupled Lorenz oscillators as described in Fig. 2.1 with the only difference is that here the parameter 28 of oscillator 1 is changed to be 27.9, while that of oscillator 2 becomes 28.1. The picture shows that, after initial contraction, the trajectories of the two oscillators level off and do not converge to 0, as opposed to the case shown in Fig. **??**.

It is known that parameter mismatch among the individual oscillators can cause bursts due to the instability of typical periodic orbits embedded in the synchronized chaotic attractor (ROH04); even within a stable region where no bubbling will occur, the states of different units will still not approach exactly the same function of time, but instead come close to each other within a neighborhood of the identical synchronization state (ROH04). This phenomena was first reported in (PC90) for two coupled Lorenz oscillators, where the variations of individual units from the identical synchronization manifold was found to scale linearly with respect to the magnitude of parameter mismatch when the mismatch is small. In (ROH04), a vari-

ational equation analogous to our Eq. (2.23) was used to study the progressive loss of synchronization stability due to bursting, which is also a relevant and interesting phenomenon. In this section we show how the master stability framework can be generalized for systems with near-identical parameters and derive stability conditions for stable near-synchronization. Again, we emphasize that the graphs under consideration are undirected as described by us in (SBN09d), while theory for directed networks are similar, and can be found by our recent paper (SBN09c).

2.3.2 Near Synchronous State (NSS)

Assume that the parameters μ_i in Eq. (2.1) are close to each other and do not change with time. Let the *average parameter* be $\bar{\mu} \equiv \frac{1}{n}\sum_{i=1}^{n}\mu_i$ and the *parameter mismatch* be $\delta\mu_i \equiv \mu_i - \bar{\mu}$. With appropriate choices of coupling strength g and network structure L, the system can have a *near synchronous state* (NSS) in which $\max_{i,j}||w_i(t) - w_j(t)|| \leq c$ as t $\to \infty$ for some small constant $c \geq 0$. When the system undergoes such near-synchronization, the trajectories of individual units are well approximated by the *average trajectory* $\bar{w} \equiv \frac{1}{n}\sum_{i=1}^{n} w_i$, which is governed by

$$\dot{\bar{w}} = \frac{1}{n}\sum_{i=1}^{n}\dot{w}_i = \frac{1}{n}\sum_{i=1}^{n} f(w_i, \mu_i), \tag{2.20}$$

With this equation, we can discuss dynamics of the bulk, or coarse scale behavior.

2.3.3 Inhomogeneity in the Variational Equations

Define the variation on each individual unit to be $\eta_i \equiv w_i - \bar{w}$ for $i = 1, 2, ..., n$. The variational equations is then

$$\begin{aligned}\dot{\eta}_i &= \Big[f(\bar{w} + \eta_i, \bar{\mu} + \delta\mu_i) - \frac{1}{n}\sum_{j=1}^{n} f(\bar{w} + \eta_j, \bar{\mu} + \delta\mu_j)\Big] \\ &\quad -\sigma\sum_{j=1}^{n} l_{ij}h(\bar{w} + \eta_j).\end{aligned} \tag{2.21}$$

Assuming that the variations η_i and the parameter mismatch $\delta\mu_i$ are small, we expand around $\bar{w}$ and $\bar{\mu}$ to obtain

$$\begin{aligned}\dot{\eta}_i &= D_w f(\bar{w},\bar{\mu})\eta_i - \sigma\sum_{j=1}^{n} l_{ij} Dh(\bar{w})\eta_j \\ &\quad + D_\mu f(\bar{w},\bar{\mu})\delta\mu_i. \end{aligned} \tag{2.22}$$

We have used $\sum_{j=1}^{n}\eta_j \equiv \sum_{j=1}^{n} w_j - n\cdot\bar{w} = 0$ and $\sum_{j=1}^{n}\delta\mu_j \equiv \sum_{j=1}^{n}\mu_j - n\cdot\bar{\mu} = 0$ in the derivation. Putting all the η_i and $\delta\mu_i$ in column vectors $\boldsymbol{\eta}$ and $\boldsymbol{\delta\mu}$, respectively, and omitting the arguments $(\bar{w},\bar{\mu})$ for simplicity, we obtain the *variational equation for the NSS*:

$$\dot{\boldsymbol{\eta}} = \Big[I_n \otimes D_w f - \sigma\cdot L\otimes Dh\Big]\boldsymbol{\eta} + \Big[I_n\otimes D_\mu f\Big]\boldsymbol{\delta\mu}. \tag{2.23}$$

When all the parameters μ_i are the same, the second term in the Eq. (2.23) disappears, and what is left is a homogeneous ODE system for $\boldsymbol{\eta}$, which may be diagonalized to obtain an equation analogous to the well-known master stability equation (PC98).

We now focus on the case in which, if there were no parameter mismatch, the system would undergo stable identical synchronization, i.e., the variation $\boldsymbol{\eta}$ would go to zero asymptotically. This situation occurs if the system represented by f, h, L and σ are in the stable regime (PC98). Because of the inhomogeneous part $\big[I_n\otimes D_\mu f\big]\boldsymbol{\delta\mu}$ due to parameter mismatch, the variational system (2.23) in general may not be asymptotically stable. We will show, however, that when the parameter mismatch is small, there may exist a NSS where $\boldsymbol{\eta}$ stays close (although not equalling) to zero. Indeed, we will show that the variational system is stable (i.e. the solution $\boldsymbol{\eta}$ is bounded as $t\to\infty$) and the bound for the solution depends linearly on the norm of the parameter mismatch $\boldsymbol{\delta\mu}$.

2.3.4 Generalized Master Stability Equations and Functions

We may uncouple the variational equation as in the identical case by diagonalizing the graph Laplacian L: $L = P\Lambda P^T$ for some orthonormal matrix P. Making the change of variable $\boldsymbol{\zeta} = (P^T\otimes I_m)\boldsymbol{\eta}$, we obtain

$$\dot{\boldsymbol{\zeta}} = \Big[I_n\otimes D_w f - \sigma\cdot\Lambda\otimes Dh\Big]\boldsymbol{\zeta} + \Big[P^T\otimes D_\mu f\Big]\boldsymbol{\delta\mu}. \tag{2.24}$$

The homogeneous part in Eq. (2.24) has block diagonal structure and we may write for each eigenmode $i \geq 2$

$$\dot{\zeta}_i = \Big[D_w f - \sigma\lambda_i DH \Big] \zeta_i + D_\mu f \cdot \sum_{j=1}^{n} u_{ij} \delta\mu_j, \tag{2.25}$$

where u_{ij} is the jth component of the ith eigenvector of L, i.e., $u_{ij} = p_{ji}$. The vector $\sum_{j=1}^{n} u_{ij}\delta\mu_j$ is the weighted average of parameter mismatch vectors, with the weights given by the components of the eigenvector associated with λ_i. It may also be thought of as an inner product of the parameter mismatch vector and the corresponding eigenvector.

From Eq. (2.25), we define a *generalized master stability equation* for nearly identical coupled dynamical systems:

$$\dot{\xi} = \Big[D_w f - \alpha \cdot DH \Big] \xi + D_\mu f \cdot \psi, \tag{2.26}$$

where we have introduced two auxiliary parameters, a scalar α and $\psi \in \Re^p$. Once the stability of Eq. 2.26 is determined as a function of α and ψ, the stability of the ith eigenmode can be found by simply setting $\alpha = \sigma\lambda_i$ and $\psi = \sum_{j=1}^{n} u_{ij}\delta\mu_j$. The problem is thus decomposed into two separate parts: one that depends only on the individual dynamics and the coupling function, and the other that depends only on the graph Laplacian and parameter mismatch. Note that the latter not only depends on the spectrum of L as in (PC98), but also on the combination of the eigenvectors and parameter mismatch. Thus, we have reduced the stability analysis of the original mn-dimensional problem to that of m-dimensional problem with one additional parameter, combined with an eigenproblem.

Note that to analyze the stability of the original system using the master stability equation, we need the associated average trajectory $\bar{w}$, which can only be obtained by solving the original system, and is impractical for large networks. We found, however, that in practice as we will confirm in examples below (or, if one is interested in details, should look into Chapters 4 and 5 of the thesis about shadowing) one may instead use a trajectory s of a single auxiliary *average unit*: $\dot{s} = f(s, \bar{\mu})$. If the average trajectory is near to be uniformly hyperbolic, the it can actually be shadowed by a true trajectory from the system $\dot{s} = f(s)$, a concept to be discussed in the next chapter.

The associated *generalized master stability function* (GMSF) $\Omega(\alpha, \psi)$ is then defined to be the asymptotic value of the norm of ξ as a function of α and ψ, given that α leads to asymptotic stable solution of the homogeneous part. Since P is orthonormal, we can predict the square-sum synchronization error in the original system (2.1) from $\Omega(\alpha, \psi)$:

$$\sum_{i=1}^{n} ||\eta_i(t)||^2 = \sum_{i=2}^{n} ||\zeta_i(t)||^2 \xrightarrow{t\to\infty} \sum_{i=2}^{n} \Omega(\alpha_i, \psi_i)^2, \tag{2.27}$$

where α_i and ψ_i correspond to the ith eigenmode and $||\cdot||$ denotes the Euclidean norm.

2.3.5 Conditions for Stable Synchronization

In the previous section we have derived a generic stability equation (2.26) for analyzing the stability of synchronization of coupled dynamical system (2.1). To analyze the stability, we now assume that the largest Lyapunov exponent of the synchronous trajectory associated with the homogeneous variational equation

$$\dot{\xi} = \Big[D_w f - \alpha Dh\Big]\xi \tag{2.28}$$

is negative for a given α, so that without parameter mismatch the error mode corresponding to this specific α goes to zero exponentially. In this case, the solution ξ^* of Eq. (2.28) can be written as: $\xi^*(t) = \Phi(t, 0)\xi(0)$, where $\Phi(t, \tau)$ is the fundamental transition matrix [2], satisfying

$$||\Phi(t, \tau)|| \leq \gamma e^{-\lambda(t-\tau)} \tag{2.29}$$

for $t \geq \tau$ and some finite positive constants γ and λ. We should note that in the case of generalized synchrony, the loss of stability of the invariant manifold need not proceed monotonically and uniformly in space. It is known that parameter mismatch can cause bursting due to increasing instability of embedded transversely unstable periodic orbits which cause short-time positivity of Lyapunov exponents (ROH04; KLK02), and this can be correspondingly interpreted from Eq. (2.29). Such transition has been called bubbling bifurcation (VHO$^+$96b; VHO96a) due to basin riddling.

[2] This transition matrix, as a function of two time variables t and τ, can be obtained by the *Peano-Baker series*, as long as $D_w f - \alpha Dh$ is continuous. See (Rug96) (Ch. 3, p. 40).

The solution to Eq. (2.26) can then be expressed by (Per96)

$$\xi(t) = \Phi(t,0)\xi(0) + \int_0^t \Phi(t,\tau)b(\tau)d\tau, \tag{2.30}$$

where $b(\tau) \equiv D_\mu f(s(\tau), \bar{\mu}) \cdot \psi$. Under the condition of Eq. (2.29), we can show that $\xi(t)$ given by Eq. (2.30) is bounded by the following inequality:

$$\begin{aligned} ||\xi(t)|| &\leq ||\Phi(t,0)|| \cdot ||\xi(0)|| + \int_0^t ||\Phi(t,\tau)||d\tau \cdot \sup_t ||b(t)|| \\ &\leq \gamma e^{-\lambda t}||\xi(0)|| + \frac{\gamma}{\lambda}(1 - e^{-\lambda t}) \sup_t ||b(t)|| \\ &\to \frac{\gamma}{\lambda} \sup_t ||b(t)|| \quad \text{as} \quad t \to \infty. \end{aligned} \tag{2.31}$$

Thus, the inhomogeneous master stability equation is stable, i.e., the solution to Eq. (2.26) is bounded asymptotically as long as i) the homogeneous system is exponentially stable, or equivalently, the maximal Lyapunov exponent is negative; and ii) the inhomogeneous part $b(\tau) \equiv D_\mu f(s(\tau), \bar{\mu}) \cdot \psi$ is bounded.

Eq. (2.29) and Eq. (2.30) also allow us to analyze quantitatively the magnitude of asymptotic error of a near-identical system. If the magnitude of parameter mismatch is scaled by a factor c, keeping all other parameters fixed, it follows from Eq. (2.30) that the corresponding solution will be

$$\widetilde{\xi}(t) = \Phi(t,0)\xi(0) + c\int_0^t \Phi(t,\tau)b(\tau)d\tau, \tag{2.32}$$

where $\xi(t)$ denotes the variation evolution of the original unscaled near-identical system. Now the first term of both Eq. (2.30) and Eq. (2.31) goes to zero exponentially according to Eq. (2.29), so that asymptotically we have $\tilde{\xi}(t) = c\xi(t)$, i.e., the variation is scaled by the same factor correspondingly.

The above analysis allows us to conclude that the extended master stability function, as a function of α and ψ, scales linearly with respect to ψ for fixed value of α if for that α with $\psi = 0$ the associated master stability equation have an exponentially stable solution. Applying this to the variational equations Eq. (2.25), it follows that if the mismatch pattern is fixed, with magnitude scaled by a factor $\epsilon > 0$, then the second term in Eq. (2.25) is scaled by ϵ for every i, resulting in each $||\zeta_i||$ scaled by ϵ. Thus, by applying Eq. (2.27), we conclude that the synchronization error $\sqrt{\sum_{i=1}^n ||\eta_i(t)||^2} = \sqrt{\sum_{i=2}^n ||\zeta_i(t)||^2}$ is scaled by the same factor ϵ for $t \gg 1$. This is

referred to as the *linear dependence on the magnitude of parameter mismatch*, for a fixed mismatch pattern.

2.3.6 Examples of Application

We consider each individual unit $w = [x, y, z]^T$ governed by the Lorenz equations:

$$\begin{aligned} \dot{x} &= 10(y - x), \\ \dot{y} &= x(r - z) - y, \\ \dot{z} &= xy - \frac{8}{3}z, \end{aligned} \tag{2.33}$$

with mismatch between units in the parameter r, i.e., r corresponds to μ in Eq. (2.1). So we have

$$D_w f = \begin{bmatrix} -\sigma & \sigma & 0 \\ r - z & -1 & -x \\ y & x & -\beta \end{bmatrix} \tag{2.34}$$

and $D_\mu f = [0, x, 0]^T$. The coupling function h is taken to be $h(w) = w$, so that $Dh(s) = I_3$ $(\forall s)$. With these choices of f and h, we numerically integrate Eq. (2.26) for a range of α and ψ and estimate the asymptotic norm of $\xi(t)$, which gives $\Omega(\alpha, \psi)$ shown in Fig. 2.9. As shown in Fig. 2.10 for examples of two random networks with 100 and 200 vertices, this estimated $\Omega(\alpha, \psi)$, combined with Eq. (2.27) gives fairly good predictions for the actual synchronization error in the full system (2.1). In addition, Fig. 2.10 confirms that the actual synchronization error scales linearly with the magnitude of the parameter mismatch, as predicted by our analysis.

2.3.7 Brief Conclusion

In this section we have analyzed the stability of synchronization in a network of coupled near-identical dynamical systems. We have shown that the well-known master stability approach can be extended to this general case, allowing us to solve the part of the problem that depends on the individual node dynamics, independently of the network structure and the parameter mismatch pattern over the network. We

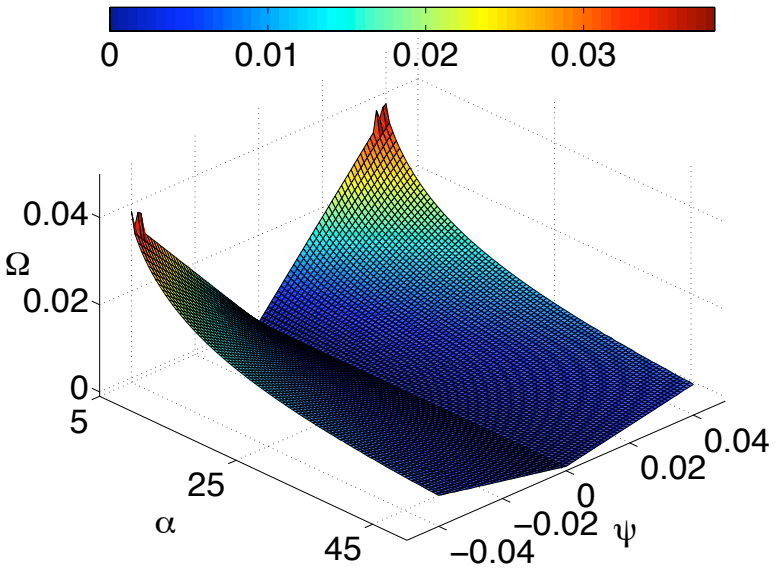

Figure 2.9. **Generalized master stability function for Lorenz equations with identity coupling.** - Density plot of the generalized master stability function $\Omega(\alpha, \psi)$ associated with arbitrary networks of near-identical Lorenz systems. It is estimated by $\sqrt{\frac{1}{T}\int_0^T ||\xi(t)||^2 dt}$ with $T = 200$ ($||.||$ denotes the Euclidean norm), where $\xi(t)$ is obtained by numerically integrating Eq. (2.28) with a time step of 0.001 and discarding initial transient. Here we have used the coupling function, $h(w) = w$.

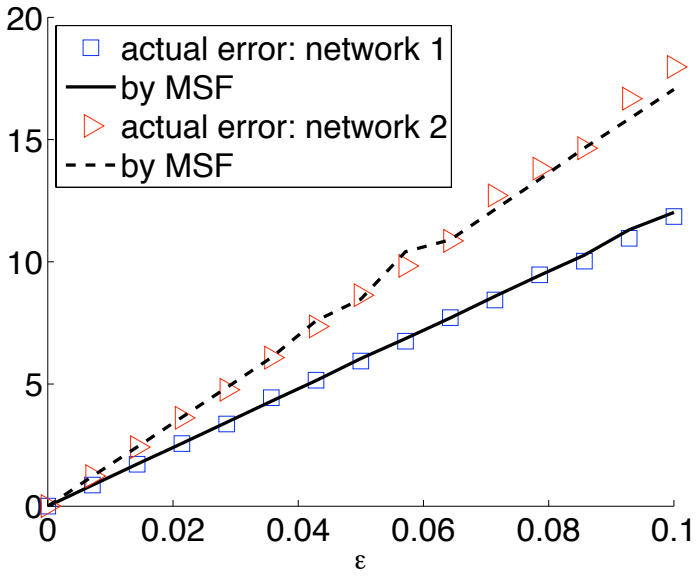

Figure 2.10. **Comparison of predicted synchronization error with actual error.** - Networks 1 is a realization of a random network consisting of 100 vertices and 513 randomly placed undirected edges, with no self loops. Likewise, network 2 is another realization with 200 vertices and 958 edges. The parameter mismatch for each network has a fixed pattern, with varying magnitude controlled by ϵ. It is generated by first choosing δ_i for each i independently from the standard Gaussian distribution, and then assigning the parameter r_i by $r_i = 28(1 + \epsilon\delta_i)$ for a given ϵ. The error prediction $\sqrt{\sum_{i=2}^{4} \Omega(\alpha_i, \psi_i)^2}$ (solid line for network 1 and dashed line for network 2) was computed using Ω displayed in Fig. **??**. Actual error (squares for network 1 and triangles for network 2) was estimated by $\sqrt{\frac{1}{T}\int_0^T \sum_{i=1}^{4} ||\eta_i(t)||^2 dt}$ with $T = 100$ computed from numerical integration of the full system (2.1) after discarding initial transient. We used $\sigma = 2$ in all calculations.

wish to point the reader to (ROH04), where mismatched oscillator synchronization is discussed and which somewhat parallels to this work. In particular, our development is in the spirit of a master stability function formalism for non-identical synchronization. We have demonstrated the validity of our analysis using a few example networks of coupled Lorenz systems. When applied to the special case of Kuramoto model (Kur84) with arbitrary network structure in the strong coupling regime, our analysis reduces to that found in (MM07). The GMSF gives simplified, accurate, and practical estimate of the magnitude of variation in a near-identical system, provided that the corresponding identical system undergoes stable synchronization according to the original MSF analysis. Furthermore, our results highlight the relevance of the Laplacian eigenvector structure, in addition to the full eigenvalue spectrum, in determining the amount of dynamical variation due to parameter mismatch among individual dynamics. This suggests that detailed knowledge of the graph structure may be important for the design of robust and reliable systems.

2.4 Application: Optimizing the Synchronization of Kuramoto Oscillators

2.4.1 Motivation and Problem Statement

Now that we have a theoretical tool (GMSF) to analyze the synchronization error when the oscillators are nearly identical, a question follows: if we are given a set of mismatched oscillators, and presumably the form of the coupling function, what is the optimal way to connect them so that the synchronization error is minimized?

If no other constraint is introduced, then a solution would be to make them *all* connected to one another, forming a *complete graph*. Such solutions are not very interesting, of course. To add a bit challenge, we further require that a graph structure is given, and all we are asked is to place the oscillators in certain ways to the nodes of the graph. Fig. 2.11 serves as a simple illustration of the problem.

Formally, in view of Eq. (2.1), the problem can be casted mathematically as:

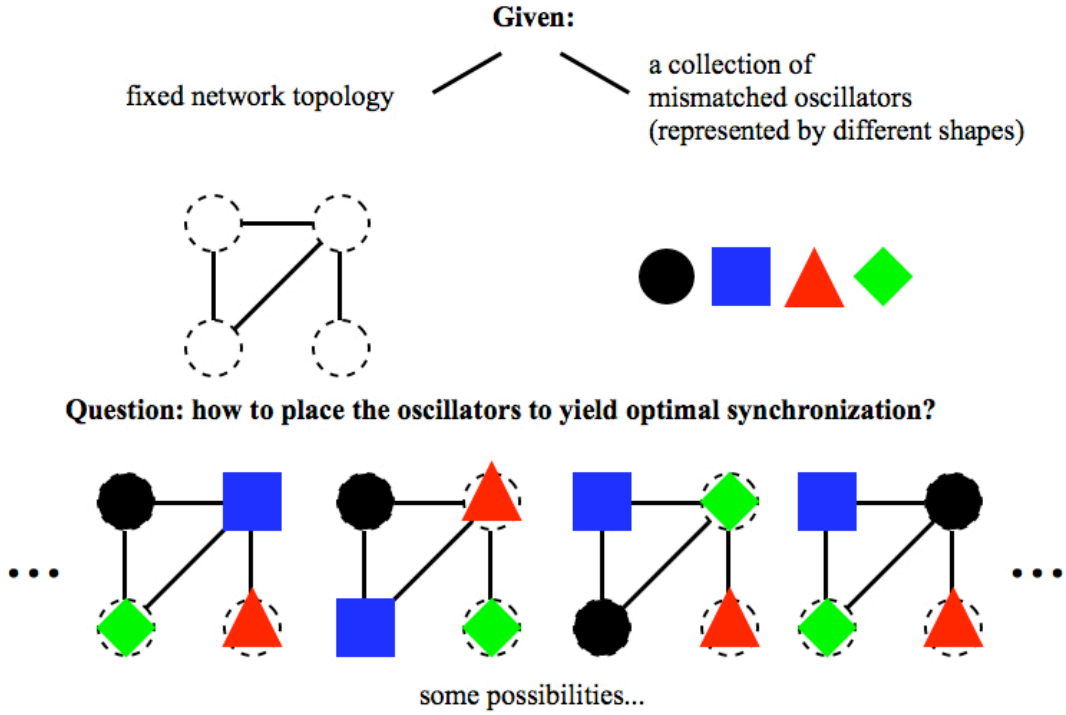

Figure 2.11. **Problem of optimizing the synchrony of mismatched oscillators for given structural constraints.** -

given matrix L, form of f and h, and a set of parameters μ_i, what is the optimal permutation $\Pi : V \to V$ such that under this permutation, the coupled system:

$$\dot{w_i} = f(w_i, \mu_{\Pi(i)}) - \sigma \sum_{j=1}^{n} l_{ij} h(w_j, \quad i = 1, 2, \ldots, n, \tag{2.35}$$

yields optimal synchronization.

Since the GMSF defines a function J for each permutation Π, the problem is thus equivalent to finding a permutation which minimizes the function value of J. Although a brute-force search is infeasible, since the possible number of permutations is huge, having defined such an appropriate measure makes it possible to find good solutions by using some optimization techniques.

2.4.2 Example of Coupled Kuramoto Oscillators

To illustrate with an example, consider in the following a system of coupled oscillators with form slightly different from our original form 2.1:

$$\dot{\theta_j} = \omega_j + \sigma \sum_{k=1}^{n} a_{kj} \sin(\theta_k - \theta_j). \tag{2.36}$$

Here θ is also known as a *phase variable* which takes value in $[0, 2\pi]$. Oscillators defined by Eq. (2.35) are known as *Kuramoto* oscillators (Kur84). To measure the synchrony, one can define parameters r and ψ by the following equation:

$$re^{i\psi} = \frac{1}{n} \sum_{j=1}^{n} e^{i\theta_j} \tag{2.37}$$

where i is the imaginary unit satisfying $i^2 = -1$. The real number r measures in some sense how coherent the oscillators evolve in time, and is usually known as the *order parameter* for such a system. Such coupled system has been studied extensively during the past twenty five years, with initial focus on the case for all to all coupling (i.e., $a_{ij} = 1$ whenever $i \neq j$) where a mean-field approach can be applied (Str00); to the study of such systems in more general networks (see ref. (ADK$^+$08) for more details).

Although there is no explicit formulae for computing r, when σ is large and the system becomes close to be coherent, Eq. (2.35) may be approximated by:

$$\begin{aligned} \dot{\theta_j} &= \omega_j + \sigma \sum_{k=1}^{n} a_{kj}(\theta_k - \theta_j) \\ &= \omega_j - \sigma \sum_{k=1}^{n} l_{kj}\theta_k \end{aligned} \tag{2.38}$$

where $L = [l_{kj}]_{n\times n}$ is defined similarly as before, which can be decomposed as: $L = P\Lambda P^T$ where the diagonals of Λ are the eigenvalues $\lambda_1, ..., \lambda_n$ of L, ordered in a nondecreasing way. For convenience, we require that $\sum_i \omega_i = 0$, which is equivalently as saying that the system is described in a rotation frame with speed of rotation equalling $\bar{\omega} \equiv \frac{1}{n}\omega_i$.

In this case, the GMSF can be computed directly through eigenvalues and eigenvectors, to yield

$$R \equiv \sum_{i}^{n} (\theta_i - \bar{\theta})^2 \xrightarrow{t\to\infty} \sum_{i=2}^{n} \Big[\frac{(P^T\omega)_i}{\lambda_i}\Big]^2 \tag{2.39}$$

given that $\lambda_i > 0$ whenever $i \geq 2$. Here $\omega = [\omega_1, ..., \omega_n]^T$, and $(P^T\omega)_i$ is the projection of ω to the $i-th$ eigenvector of L.

If we further assume without loss of generality that at time $t = 0$, $\sum_i \theta_i = 0$, then the quantity R relate to r in the following way, approximately, when $\theta_1 \approx \theta_2 \approx ... \approx \theta_n \approx 0$. Assuming that $\theta \equiv \max_i |\theta_i| << 1$, we have:

$$\begin{aligned} r^2 &= \frac{1}{n^2}\big[(\cos\theta_1 + ... + \cos\theta_n)^2 + (\sin\theta_1 + ... + \sin\theta_n)^2\big] \\ &\approx \frac{1}{n^2}\big[(n - \frac{1}{2}[{\theta_1}^2 + ... + {\theta_n}^2])^2 + (\theta_1 + ... + \theta_n)^2\big] \\ &\approx \frac{1}{n^2}\big[n^2 - ({\theta_1}^2 + ... + {\theta_n}^2)\big] \\ &= \frac{1}{n^2}(1 - R^2). \end{aligned} \tag{2.40}$$

Thus, maximizing r is equivalent as minimizing R in this case.

Since no better measure that is available analytically for relating the synchrony with the distribution ω and network topology, we will use R as a our cost function as a reasonable choice. Note that when possible permutation is allowed, the function R

will depend on the permutation matrix Π, for given ω and L, as:

$$R(\Pi) \equiv \sum_{i=2}^{n} \left[\frac{(P^T \Pi \omega)_i}{\lambda_i} \right]^2. \tag{2.41}$$

Solving Eq. (2.41) is in general infeasible. We here adopt an optimization technique called simulated annealing (Fis05) to find approximate solutions that are reasonable.

To illustrate by an example, consider a collection of Kuramoto oscillators with frequency ω_i distributed independently according to the Gaussian distribution $N(0, (\frac{\pi}{10})^2)$. The underlying graph is a random graph with $n = 100$ nodes and $m = 267$ randomly placed edges. In Fig. 2.12 we show the minimum value of R found after t steps as a function of t, the number of steps in the simulated annealing process. We take the output of this algorithm to be an approximately optimal solution, and compare its order parameter to other configurations, shown in Fig. 2.13.

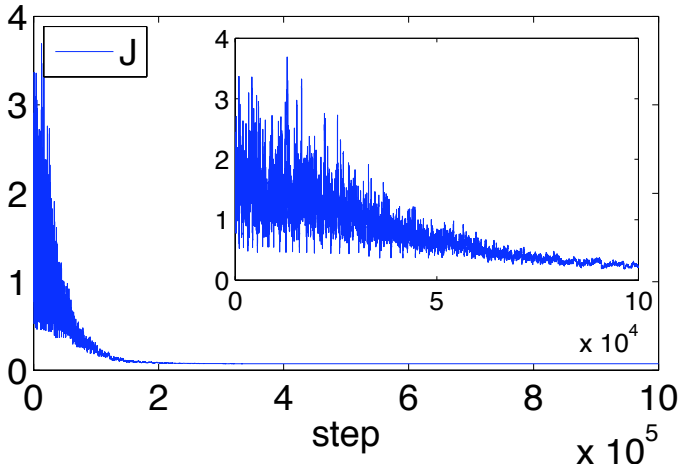

Figure 2.12. **Evolution of the cost function for Kuramoto oscillators using simulated annealing.** - The algorithm was terminated after 10^5 steps, by when there seems to be no decrease of the function value. The inset shows the function value for the first 10^4 steps. Details of implementing the simulated annealing algorithm include appropriate choice of initial temperature, annealing process, and stopping criteria, which is not discussed in this thesis due to the limitation of space.

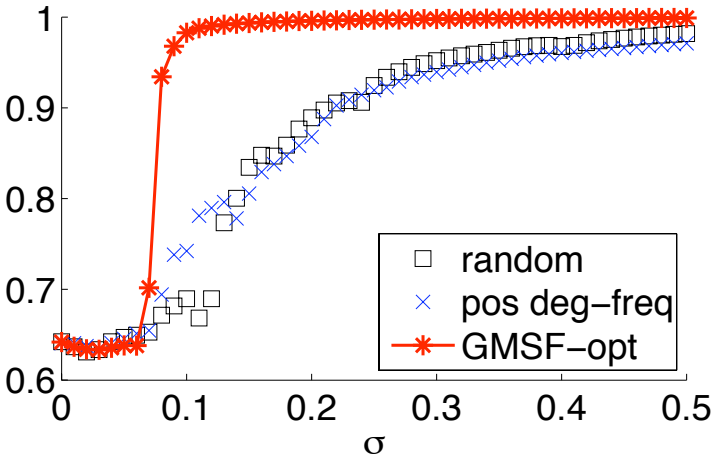

Figure 2.13. **Comparing the order parameter for different configurations of Kuramoto oscillators.** - The curves show the order parameter r, as defined in Eq. (2.37), as a function of the coupling strength σ when the frequencies ω_i are distributed either *randomly* (black squares) or with positive degree-frequency correlation (blue crosses), or according to the optimal output of simulated annealing algorithm (red stars). The curves are obtained by an average of 100 realizations with different initial conditions, and r is computed by taking the time average of $t = 50$ to $t = 100$.

Chapter 3

Network of Networks: Multi-scale Dynamics on Networks

3.1 Multi-scale Dynamics on Networks

When a system exhibits complicated spatial-temporal dynamics, it is often desirable to reduce the dimension of the problem, if possible. This idea of dimensionality reduction has been found in other branches in science, for example, the principle component analysis based on singular value decomposition; or clustering of data points (DHS00). When a system is composed of coupled oscillators interacting with each other in nontrivial ways, it is natural to ask, how can we apply coarse-grain analysis to extract a simplified model while maintaining useful information about the whole system? Fig. 3.1 shows an example of such a problem. Suppose that there are dynamics on the network shown in the picture. It looks complicated in the finest scale; however, in a somehow median scale we see six clusters interacting with each other. The main object of Chapters 3, 4, and 5 is to quantify, when dynamics is introduced on some nontrivial network topology, reduced order models, and how to judge quality of such a model.

In this Chapter (Chapter 4) we develop theory for analyzing problems of multi-scale modeling of network dynamics. Consider the following differential equations

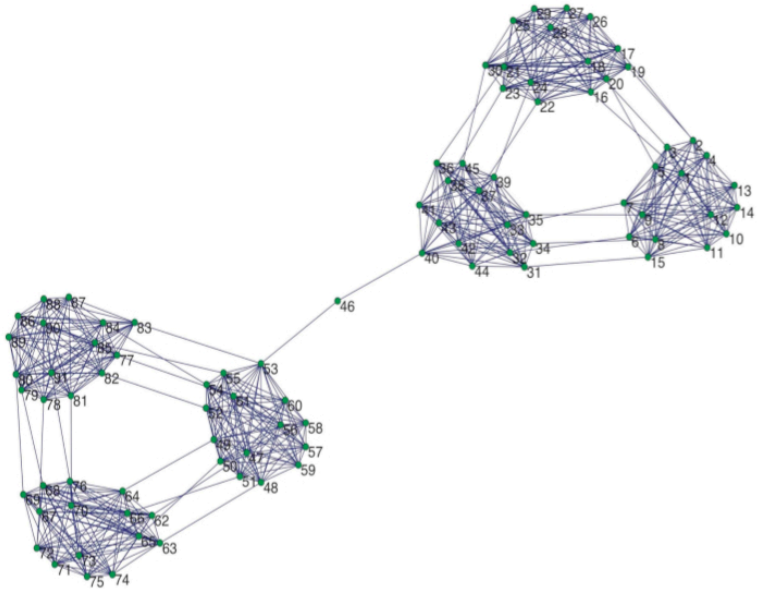

Figure 3.1. **Illustration of the problem of multi-scale modeling for network dynamics.** - A coupled oscillator network may exhibit hierarchical scales depending on both network topology and dynamics on individual nodes. In the picture it is easy to see clusters. However, in general it may require a complicated computational approach to reveal such clustering, and when dynamics is considered, understanding the interplay between the network topology and heterogeneity of individual dynamics is a bit challenging (Bol06).

which describes an *oscillator network (OSN)*:

$$\dot{\theta}_i = f_i(\theta_i) + \sigma \sum_{j=1}^{n} a_{ij} h(\theta_j - \theta_i), \tag{3.1}$$

here f_i are the individual dynamics, $h(x)$ is a coupling function, $\sigma > 0$ is the coupling strength, and the network is characterized by the adjacency matrix $A = [a_{ij}]_{n \times n}$.

For example, if we use Kuramoto oscillators coupled through a network shown in Fig. 3.2, the time series seem to form clusters which coincide with the network structure. If we zoom in further, a finer clustered structure is revealed from the time series, which correspond to a finer partition of the network. How to model those coarse scale dynamics? How to define an appropriate average among oscillators in a network? Such questions will be investigated in the next few sections.

3.2 Coarse-grain Modeling of Multi-scale Dynamics on a Network

To define a reduced order model for network dynamics, it is crucial to have an appropriate partition of the nodes into groups where each group represent a collection of oscillators which are nearly coherent. The partition of nodes in a network which results in different groups is often referred to as, in the language of complex network study, problem of *finding communities*[1], where the groups are phrased as *communities*. This type of clustering of nodes were first used in social network research, and in recent years studied a lot for large networks by statistical physicists due to the seminal work by Newman (New06). See (POM09) as a beautiful modern introduction to this topic, and the references therein for more details. This concept of communities naturally lead to the coarse-graining of networks, studied in (Kim04) for a model introduced in (RCb02); in (MGK06) to develop computational tool for simulate coupled oscillators with fewer variables; in (GD07; GD08) where spectral coarse-grain was introduced to group nodes with respect to their similarity in eigenvector components; and very recently in (PKJ09) for biological feedback networks.

[1] In the study of complex networks, communities are often referred to (in a non rigorous sense) as densely connected groups of nodes.

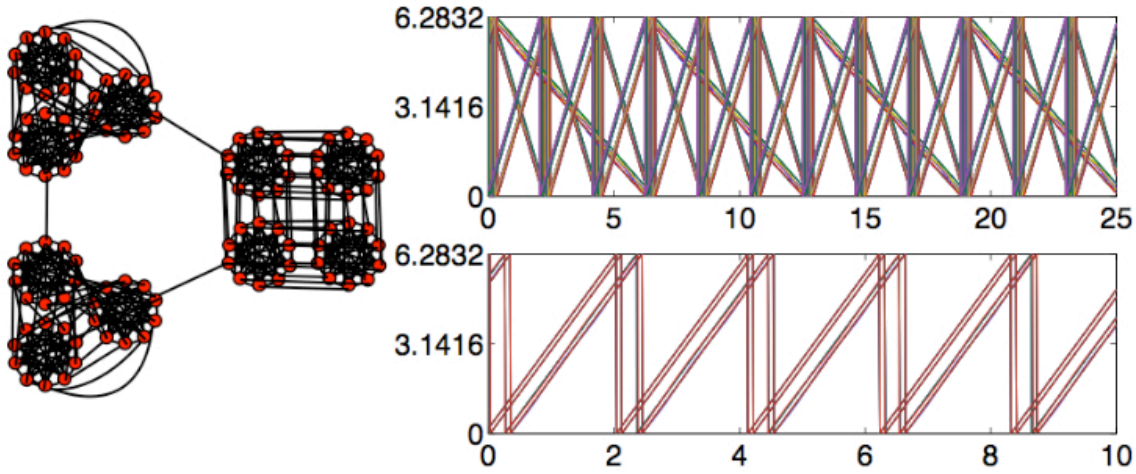

Figure 3.2. **Motivation of multi-scale dynamics on networks.** - In the left we show a network which exhibits multi-scale topological structure. The network is constructed as follows: first construct 3 complete graphs each consist of 10 nodes, label them from 1 to 30; then, connect nodes i to nodes $10+i$ and $20+i$ for each $i \in \{1,\ldots,10\}$, also connect for those i, nodes $10+i$ to $20+i$, this corresponds to the triangle like part in the upper left of the network shown in the figure. Make a copy of this network (lower left) and connect these two networks by one edge. Then, by similar process, generate a subgraph of 40 nodes (right part of the network), and connect with each of the 30 nodes subgraph with one edge. The Kuramoto oscillators follow Eq. (3.1) where θ_i is a phase variable ($\theta_i \in [0, 2\pi]$) $f_i = \omega_i$ are the natural frequencies of each oscillator, and $h(x) = \sin(x)$. Here the natural frequencies of the oscillators are chosen in such a way that in the upper left group have natural frequencies $[-3.25 * ones(1,10), -3 * ones(1,10), -2.75 * ones(1,10)]$; the lower left group: $[-1.25 * ones(1,10), -1 * ones(1,10), -0.75 * ones(1,10)]$; and the middle right group: $[2.4 * ones(1,10), 2.8 * ones(1,10), 3.2 * ones(1,10), 3.6 * ones(1,10)]$. The coupling strength is chosen as $\sigma = 0.5$. The right part of this figure shows, on the top, time series of all the oscillators; and on the bottom, time series of the oscillators from the 40 node square-like subgraph.

For a graph $G = (V, E)$, a partition of the network into groups, or communities is analogous to a quotient operation on the vertex set of a graph, and can be characterized by a *projection function* $P : V \to \tilde{V}$, which defines an equivalence relation between elements in V. By convention, the elements for the set V and $\tilde{V}$ are represented by natural numbers, so that $V = \{1, 2, \ldots, n\}$ and $\tilde{V} = \{1, \ldots, K\}$ where $1 \leq K \leq n$. The projection of a node i under P is denoted by P_i. Two nodes i and j in V belong to the same *group* ℓ if they belong to the same equivalence class under the projection P, or more precisely, $P_i = P_j = \ell$.

Define, for each $\ell \in \{1, \ldots, K\}$, $C_\ell \equiv \{i : P(i) = \ell\}$, i.e., C_ℓ is the set of nodes belonging to group ℓ. Using this notation, by rearranging the sum from 1 to n, the equation Eq. (3.1) for an OSN may be written as:

$$\begin{aligned} \dot{\theta}_i &= f_i(\theta_i) + \sigma \sum_{j=1}^{n} a_{ij} h(\theta_j - \theta_i) \\ &= f_i(\theta_i) + \sigma \sum_{k=1}^{K} \sum_{j \in C_k} a_{ij} h(\theta_j - \theta_i). \end{aligned} \tag{3.2}$$

When there is strong coherence between oscillators belonging to the same groups, the following coarse-grained oscillators will be useful. Define, for each $\ell \in \tilde{V}$,

$$\phi_\ell \equiv \frac{1}{|C_\ell|} \sum_{i \in C_\ell} \theta_i \tag{3.3}$$

Then, the equation governing ϕ_ℓ is simply:

$$\dot{\phi}_\ell = \frac{1}{|C_\ell|} \sum_{i \in C_\ell} f_i(\theta_i) + \sigma \sum_{j=1}^{n} \frac{1}{|C_\ell|} \sum_{i \in C_\ell} a_{ij} h(\theta_j - \theta_i). \tag{3.4}$$

With the assumption that $\theta_i \approx \phi_{P_i}$ for each $i \in V$, we replace each θ_i in the above equation with a coarse variable ϕ_{P_i}, and then replace all the ϕ by ψ to differ from the equation that exactly governs the dynamics of ϕ. This leads to the following equation, for $\ell \in \tilde{V}$:

$$\begin{aligned} \dot{\psi}_\ell &= \frac{1}{|C_\ell|} \sum_{i \in C_\ell} f_i(\psi_\ell) + \sigma \sum_{j=1}^{n} \frac{1}{|C_\ell|} \sum_{i \in C_\ell} a_{ij} h(\psi_{P_j} - \psi_{P_i}) \\ &= \frac{1}{|C_\ell|} \sum_{i \in C_\ell} f_i(\psi_\ell) + \sigma \sum_{k=1}^{K} \frac{1}{|C_\ell|} \sum_{i \in C_\ell} \sum_{j \in C_k} a_{ij} h(\psi_{P_j} - \psi_{P_i}) \qquad (3.5) \\ &= g_\ell(\psi_\ell) + \sum_{k=1}^{K} b_{\ell k} h(\psi_k - \psi_\ell), \qquad (3.6) \end{aligned}$$

where we have defined, for convenience,

$$g_\ell(\psi_\ell) \equiv \frac{1}{|C_\ell|} \sum_{i \in C_\ell} f_i(\psi_\ell) \tag{3.7}$$

and

$$b_{\ell k} \equiv \frac{1}{|C_\ell|} \sum_{i \in C_\ell} \sum_{j \in C_k} a_{ij}. \tag{3.8}$$

Notice that Eq. (3.6) has exactly the same form as Eq. (3.2), except that the f replaced by g, a_{ij} replaced by $b_{\ell k}$, and the indices run in a different range. Thus, Eq. (3.6) defines another oscillator network, with fewer nodes ($k = 1, \ldots, K$). Eq. (3.6) will be referred to as the *coarse-grained oscillator network (CGOSN)* (with respect to the projection function P) of the original OSN. The *dynamics* of this CGOSN is usually similar as the individual dynamics f_i if all the f_i are close to each other in some sense; and the $b_{\ell k}$ defines an adjacency matrix $B = [b_{\ell k}]_{K \times K}$ for a network consisting of K nodes, each represent a group of the original network characterized by A.

Since we will use Kuramoto oscillators for most of the examples for this section, it is convenient to write the above equations in the specific form of coupled Kuramoto oscillators, in which case Eq. (3.2) becomes:

$$\dot{\theta}_i = \omega_i + \sigma \sum_{j=1}^{n} a_{ij} \sin(\theta_j - \theta_i) = \omega_i + \sigma \sum_{k=1}^{K} \sum_{j \in C_k} a_{ij} \sin(\theta_j - \theta_i), \tag{3.9}$$

where θ_i is the *phase* of oscillator i, $\theta_i \in [0, 2\pi]$; ω_i is the natural frequency of oscillator i, and the rest can be read from the equation. The average oscillator of a group C_ℓ defined by Eq. (3.3) has its own dynamics (analogous to Eq. (3.4)):

$$\dot{\phi}_\ell = \frac{1}{|C_\ell|} \sum_{i \in C_\ell} \omega_i + \sigma \sum_{j=1}^{n} \frac{1}{|C_\ell|} \sum_{i \in C_\ell} a_{ij} \sin(\theta_j - \theta_i). \tag{3.10}$$

The corresponding CGOSN is simply described by, for each $\ell \in \{1, \ldots, K\}$,

$$\begin{aligned} \dot{\psi}_\ell &= \frac{1}{|C_\ell|} \sum_{i \in C_\ell} \omega_i + \sigma \sum_{j=1}^{n} \frac{1}{|C_\ell|} \sum_{i \in C_\ell} a_{ij} \sin(\psi_{P_j} - \psi_{P_i}) \\ &= g_\ell(\psi_\ell) + \sum_{k=1}^{K} b_{\ell k} h(\psi_k - \psi_\ell), \end{aligned} \tag{3.11}$$

where

$$g_\ell(\psi_\ell) \equiv \frac{1}{|C_\ell|}\sum_{i\in C_\ell}\omega_i, \quad b_{\ell k} \equiv \frac{1}{|C_\ell|}\sum_{i\in C_\ell}\sum_{j\in C_k} a_{ij}. \tag{3.12}$$

For example, the network shown in Fig. 3.2 has a clear *community structure*, and it produces time series which seem to form three distinct clusters. If we partition the network such that $\tilde{V} = \{1, 2, 3\}$, and

$$C_1 = \{1, \ldots, 30\}, C_2 = \{31, \ldots, 60\}, C_3 = \{61, \ldots, 100\}, \tag{3.13}$$

Then the coarse scale network (described by B in Eq. (3.6)) can be obtained as

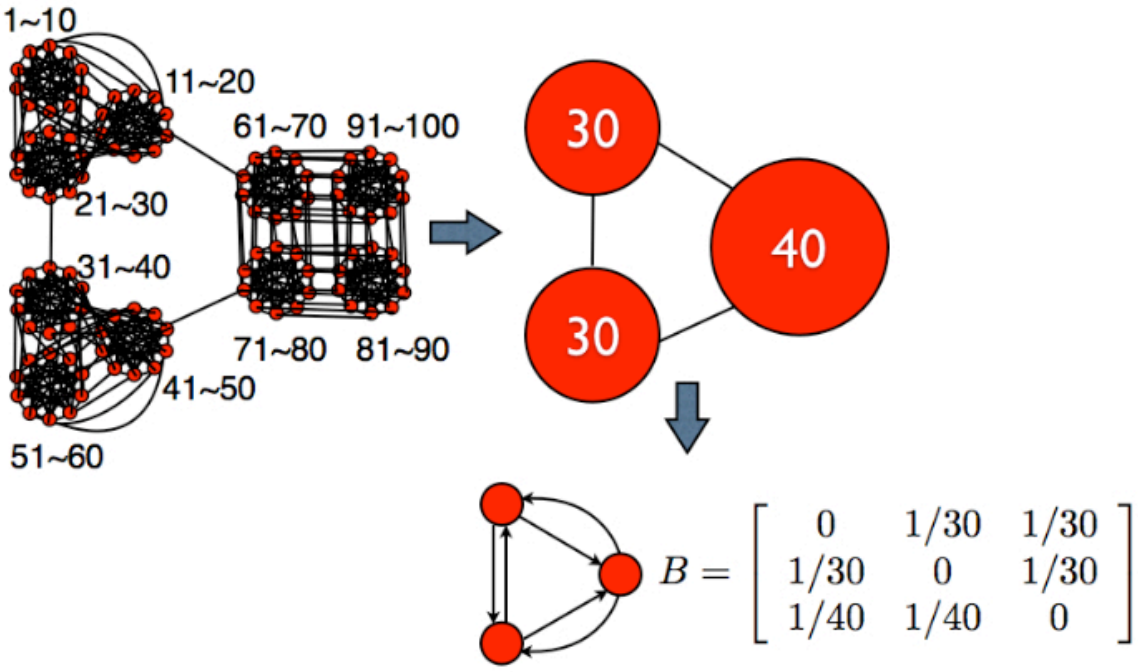

Figure 3.3. **Partition into groups of a benchmark OSN and its resulting CGOSN.** - In this example we partition the nodes of the original OSN (left) into three groups, shown in the middle, where the white numbers 30, 30, and 40 indicate the number of nodes in the three groups. Below we show the resulting CGOSN, where each node represent an average of a group of nodes, and the weight on an edge (ℓ, k) in this coarse-grained network equals the average number of links from group ℓ to group k. The matrix beside the network is the adjacency matrix for the coarse-grained network.

the process shown in Fig. 3.3: first sum over the nodes in each group, then average; the weight on an edge (ℓ, k) in B equals the number of edges from group ℓ to group k, divided by the total number of nodes in group ℓ. Thus, even when the original

network A is undirected, this coarse scale network B can in general be a directed network. Once the coarse scale network is found, the corresponding dynamics can be obtained according to Eq. (3.6).

In Fig. 3.4 we compare the dynamics from the original OSN for the network shown in Fig. 3.3, its average time series of each group, and the time series obtained from the dynamics of the corresponding CGOSN, respectively. The CGOSN provides a reduced order model, which perfectly captures the coarse scale behavior of the original network dynamics.

The projection of a given OSN need not be unique. For example, for the network dynamics shown in Fig. 3.3 and Fig. 3.4, we could have partitioned the nodes into 10 distinct groups, instead of three. The two different partitions are shown in Fig. 3.5. Similarly as the comparison shown in Fig. 3.4, we show time series generated by the CGOSN corresponding to the partition into 9 groups in Fig. 3.6. Again, the CGOSN successfully reduce the complexity of the original model, and in the meanwhile produces time series that match with the ones from the original OSN.

3.3 Uncovering Spatial Scale by Time Series? A Counter Example to Popular Intuition.

A common belief when performing time series analysis in the field of complex networks is that, correlation between two time series indicate a strong connection between the two sources who generate them. Thus, clustering by time series shall lead to communities in a network, which provides a way to uncover topological scale of a network by simply looking at some time series generated from some coupled dynamics run on the network (ADP06). In this section, we show a counter example where clustering of time series provides totally different result as one might guess from the network structure.

Consider a network of Kuramoto oscillators as described by Eq. , where the network structure is shown in the upper part of Fig. 3.7. Here the network consists of two complete subgraphs (each of 10 nodes) with two edges connecting in between.

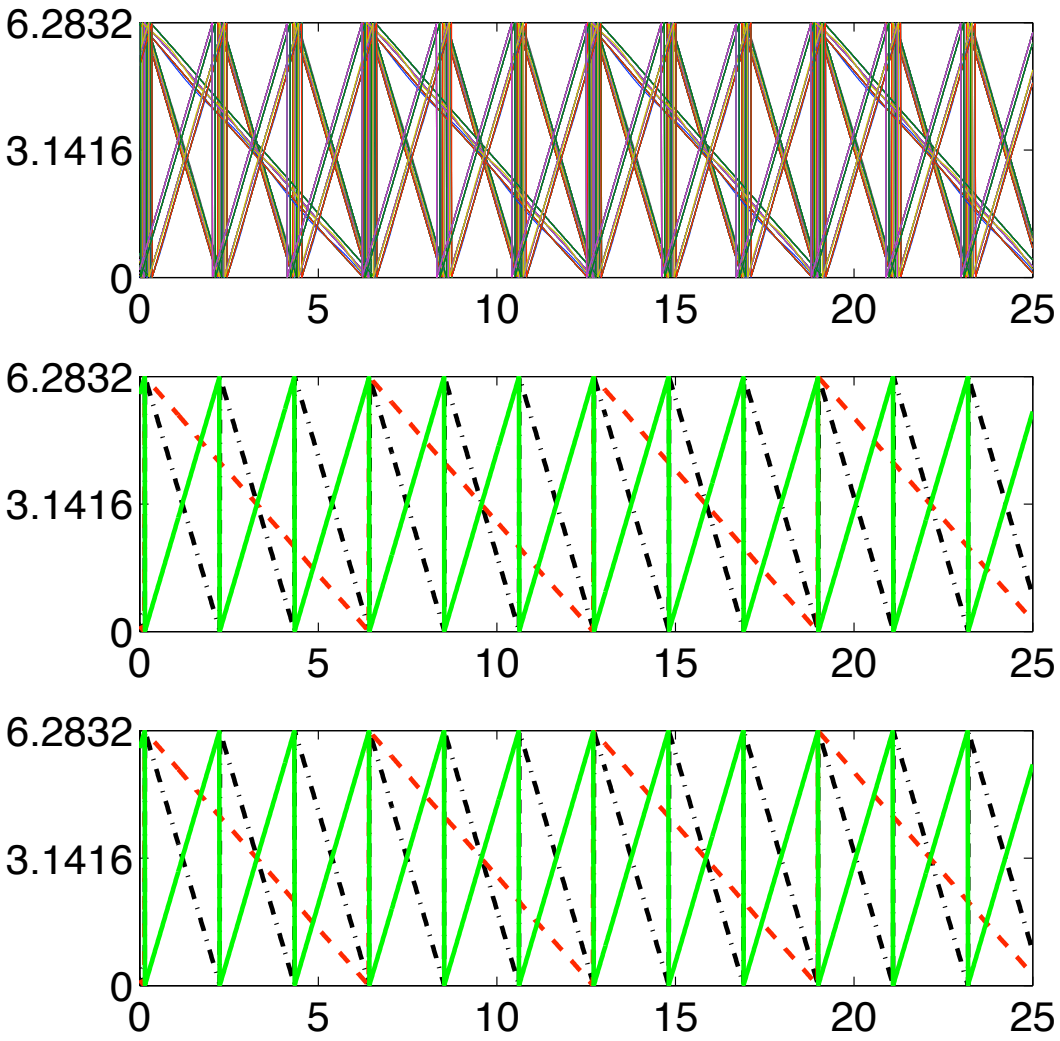

Figure 3.4. **Multiscale Kuramoto time series.** - Upper panel shows the time series of a coupled Kuramoto oscillators where the network topology is shown in the left panel in Fig. 3.5. Middle panel shows the average time series within each group. In the lower panel we show time series generated from a reduced order model where the model consists of three nodes coupled through the network shown in the right panel of Fig. 3.5.

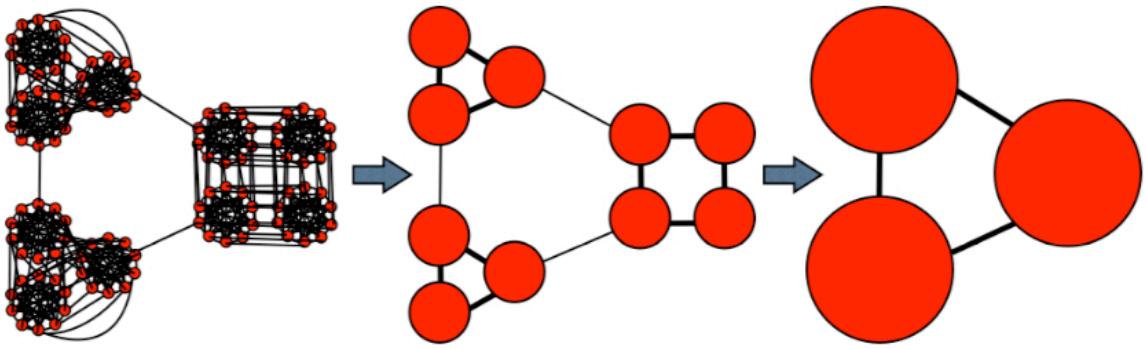

Figure 3.5. **Multiscale model of a benchmark network.** - This figure shows two different scales of the original network (on the left), which correspond to two different coarse-scale network model, one consists of 3 nodes (shown in the middle), and the other 9 nodes (shown on the right).

Indeed, what is hidden behind this surprise/puzzle is the *heterogeneity of the dynamics*. For this example, we have chosen, for all the nodes in the upper half, natural frequency -1, and for those in the lower half, natural frequency $+1$ (with coupling strength $\sigma = 0.1$). Thus, even though a complete subgraph usually forms a strong community in many situations, when dynamics is involved, for example, in this case, within each of these subgraphs there is strong heterogeneity in the individual dynamics between nodes, and this heterogeneity has dominated the homogeneity in the structure, and lead to the clustering of time series correlate with the partition according to dynamical homogeneity, instead of structural homogeneity.

Finding dynamical relevant clusters for an OSN is indeed important for the multi-scale modeling of spatio-temporal dynamics, since it is only after the knowledge of a partition can we form a CGOSN to reduce the order of complexity of the original model. However, although the dynamical clusters might be the same as what structural partitions provide, as in the examples in the previous section; in general they might have nothing to do with structural clusters, as in the example of this section. How to find those dynamical clusters when complicated dynamics is involved, as well as nontrivial network structures, would be the main goal of next section.

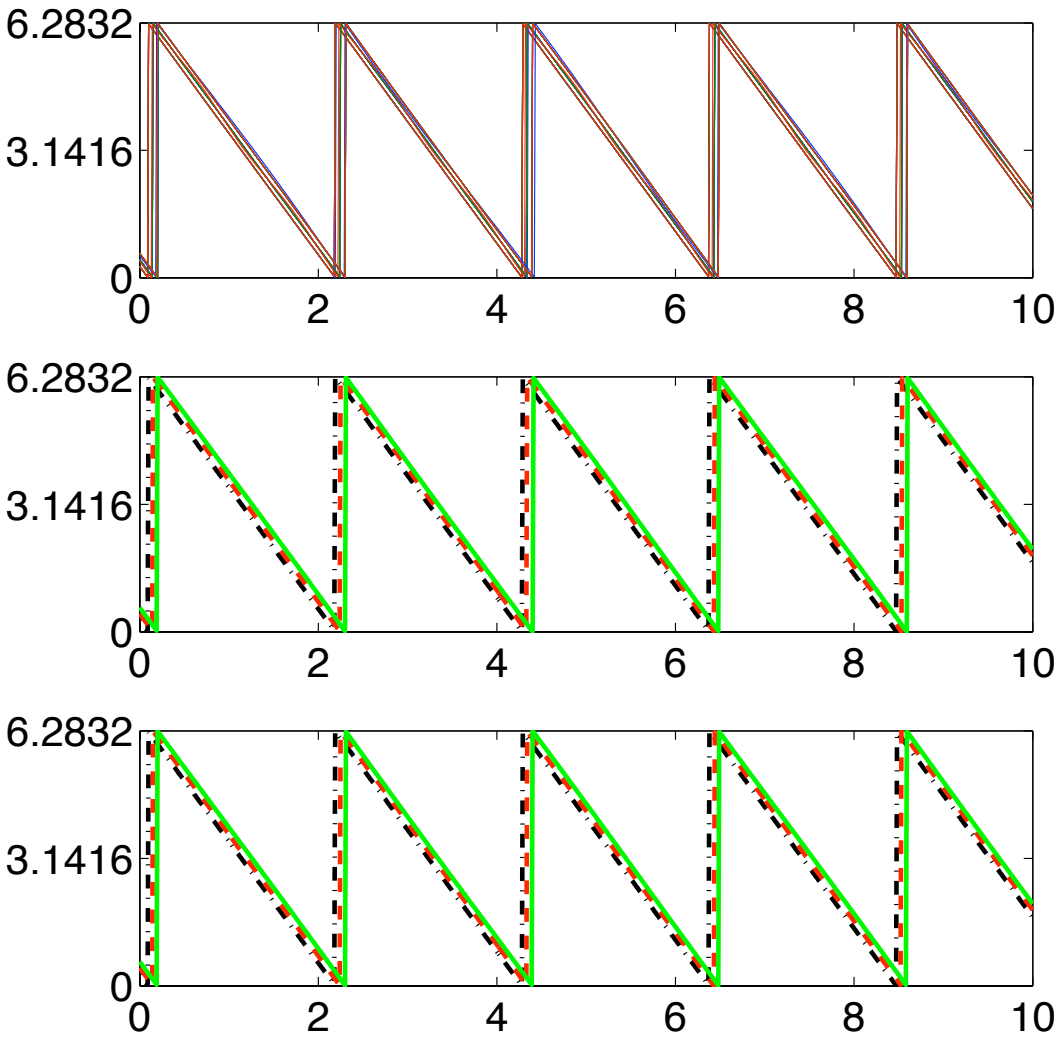

Figure 3.6. **Comparing time series from a OSN and its CGOSN, group 1.** - Comparison of time series from group 1 (the upper left group) of the network shown in the left of Fig. 3.5 and those from the upper left part of a coarse-grained network shown in the middle of Fig. 3.5. Similar pictures are observed for the other two groups, and will not be shown in the thesis.

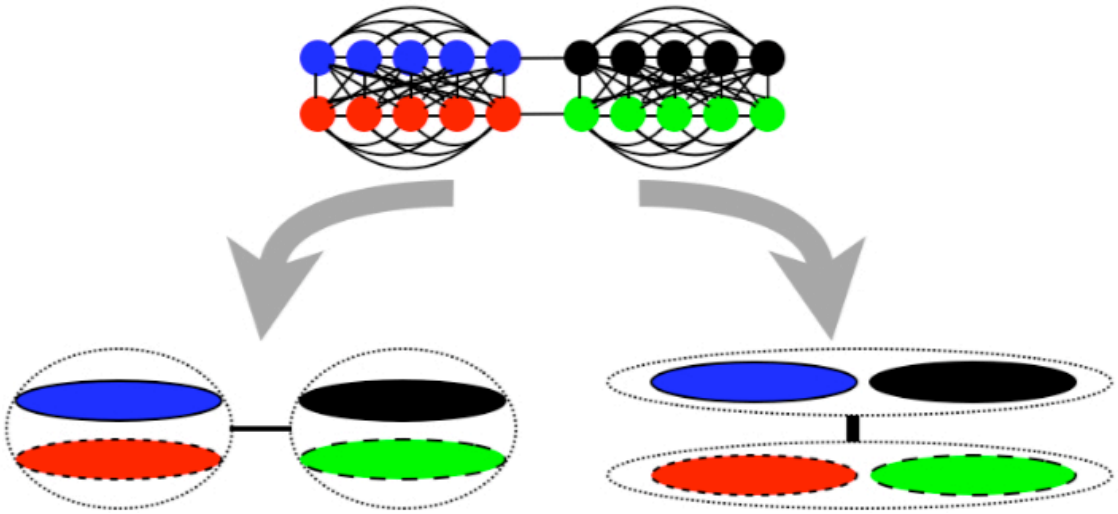

Figure 3.7. **Partition into two clusters in different ways.** - Here the network consists of two complete subgraphs (each of 10 nodes) with two edges connecting in between. A natural partition of this network is two groups where each group contains a complete subgraph; indeed, the time series as shown in Fig. 3.8 clearly shows two distinct clusters, which is a strong indication that there are two densely connected clusters in the network. However, as we will discuss later (and shown in Fig. 3.8 and Fig. 3.9), the more appropriate partition with the dynamics we have chosen on this network is the one on the right, which takes half of the nodes from each complete subgraph to form one group, the the rest the other.

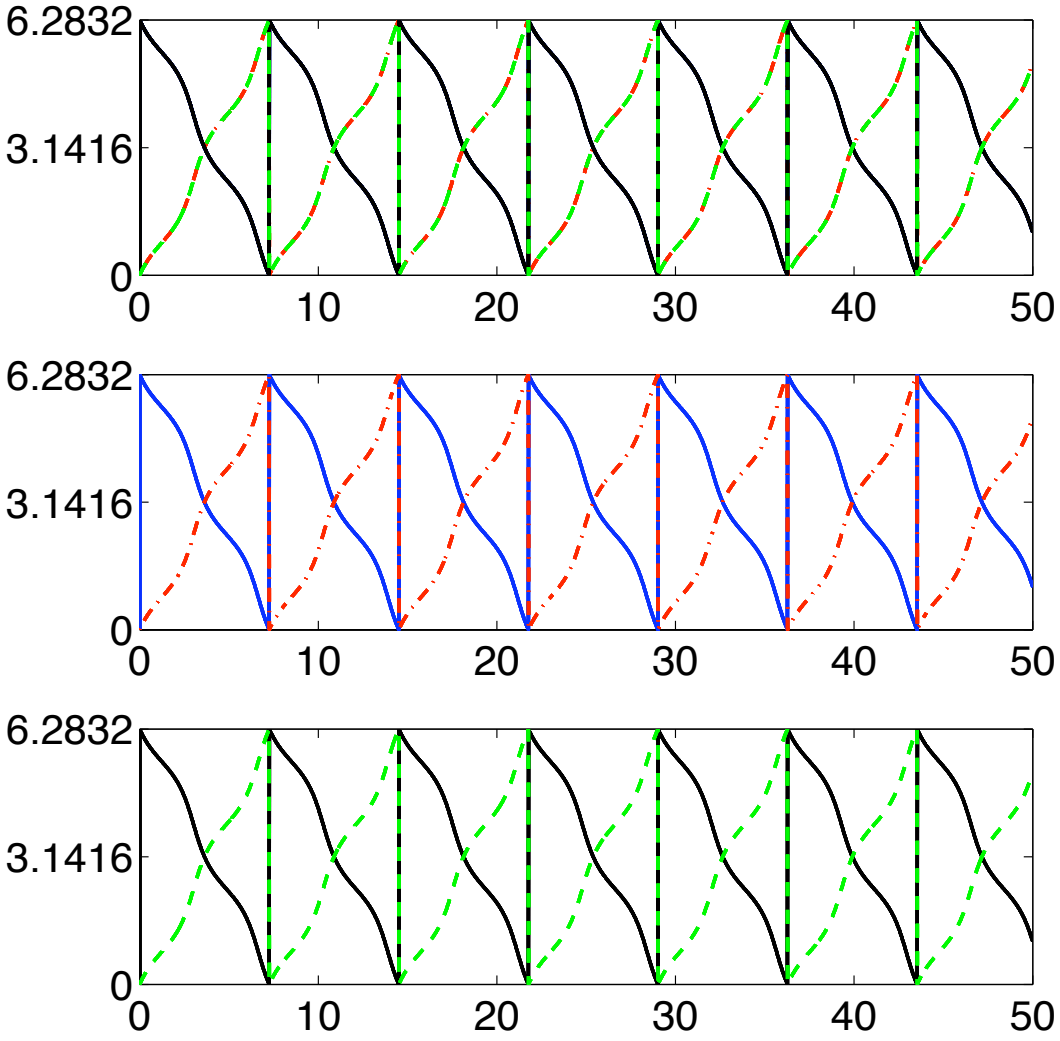

Figure 3.8. **Time series clustering vs. structural partition.** - In the upper panel we show time series of Kuramoto oscillators coupled through a network shown in Fig. 3.7. Here we use blue solid lines to represent time series from the upper half of the left complete subgraph; red dotted lines to represent those from the lower half on the left; black solid lines for the upper half on the right; and green dashed lines for the lower half on the right. The time series seem to form two clusters. On the other hand, the network has a clear structural partition shown in the lower left of Fig. 3.7. According to this partition, we show time series from group 1 in the middle panel and group 2 in the lower panel. Surprisingly, the oscillators in each group do not evolve even close as each other, but rather, seem to be uncorrelated. In this example, if we partition according to the lower right of Fig. 3.7, then the oscillators within each group do seem to follow single coarse trajectories.

3.4 Finding the Right Partition: Structural vs. Dynamical Heterogeneity

As discussed in the previous section, finding a good dynamical relevant partition is important, and non-trivial. Although a general theory has not been developed by anyone, we here will propose an empirical treatment which serves as a first step towards a deeper understanding of the problem. This treatment is based on the *variational equations* to be discussed below. We present this analysis for Kuramoto oscillators for convenience, the generalization to other systems seem to be straightforward, and will be omitted from the thesis.

Suppose that we partition a OSN using a projection function P into K distinct groups denoted by $C_1, \ldots, C_K$. If the partition is good, it shall follow that for each $i \in V$, $\theta_i \approx \phi_{P_i}$. Define

$$\xi_i \equiv \theta_i - \phi_{P_i}, \tag{3.14}$$

where ϕ_{P_i} is the average of oscillators in the group P_i, as defined before in Eq. (3.3). Then, following from Eq. (3.3) and Eq. (3.10), we have:

$$\dot{\xi}_i = \Big(\omega_i - \frac{1}{|C_\ell|}\sum_{j\in C_\ell}\omega_j\Big) + \sigma\sum_{j=1}^{n}\Big(\sum_{i\in C_\ell}\frac{a_{ij}}{|C_\ell|} - a_{ij}\Big)\sin(\theta_j - \theta_i). \tag{3.15}$$

The *dynamical heterogeneity* is characterized by the first term in the above equation, and will be denoted in the later, for oscillator i in a given OSN and a given partition, by

$$\Delta_i^D \equiv \omega_i - \frac{1}{|C_\ell|}\sum_{j\in C_\ell}\omega_j; \tag{3.16}$$

on the other hand, by a mean-field like assumption so that the term $\sin(\theta_j - \theta_i)$ is replaced by some common constant for each i, what is relevant in the second term will be purely dependent on the network structure and the partition, so the *structural heterogeneity* will be defined as:

$$\Delta_i^S \equiv \sigma\sum_{j=1}^{n}\Big(\sum_{i\in C_\ell}\frac{a_{ij}}{|C_\ell|} - a_{ij}\Big). \tag{3.17}$$

In an ideal situation where both Δ_i^D and Δ_i^S are 0 for each i, the partition is perfect

which leads to identically synchronized clusters[2], and the stability of sycnronization of each individual group/cluster can be analyzed by means of MSF discussed in Chapter 2. This will happen specifically when the OSN consists of *disconnected components* where each component consists of *identical oscillators* [a theorem can be proven for this situation, and is left as an exercise for the readers.] In general, the numbers will not be identically 0, and a partition is a good one if both $||\Delta_i^D||$ and $||\Delta_i^S||$ are small.

We next illustrate how to judge the quality of a partition by means of a cost function defined as, for a given OSN and a partition P,

$$\Delta \equiv \sum_{i=1}^{n} \left(||\Delta_i^D||^2 + ||\Delta_i^S||^2 \right). \tag{3.18}$$

To find a reasonably good partition is then equivalent as minimizing this cost function in the space of partition functions. Two examples shown in Fig. 3.9 and Fig. 3.10 confirm that the minimum of this function indeed reveals appropriate dynamical relevant partition of an OSN.

3.5 Difficulty for Coupled Chaotic Oscillators

Although the multi-scale modeling approach discussed in this chapter works well for systems such as coupled Kuramoto oscillators by direct inspection (i.e. compare time series from the original OSN and its CGOSN), whether or not similar analysis carry through for chaotic systems is a more difficult problem, and will be discussed in detail in the next two chapters. Fig. 3.11 shows an example illustrating the difficulty in judging the model quality of a reduced order model for a coupled chaotic oscillator network.

[2]In this case the *Golubitsky coloring* is equivalent as the perfect partition, where both the dynamical and structural heterogeneity becomes zero (GSBC99; SGP03). In general, when no such perfect symmetry can be found, our approach serves as a practical method to find a good, although not necessarily perfect partition of the network.

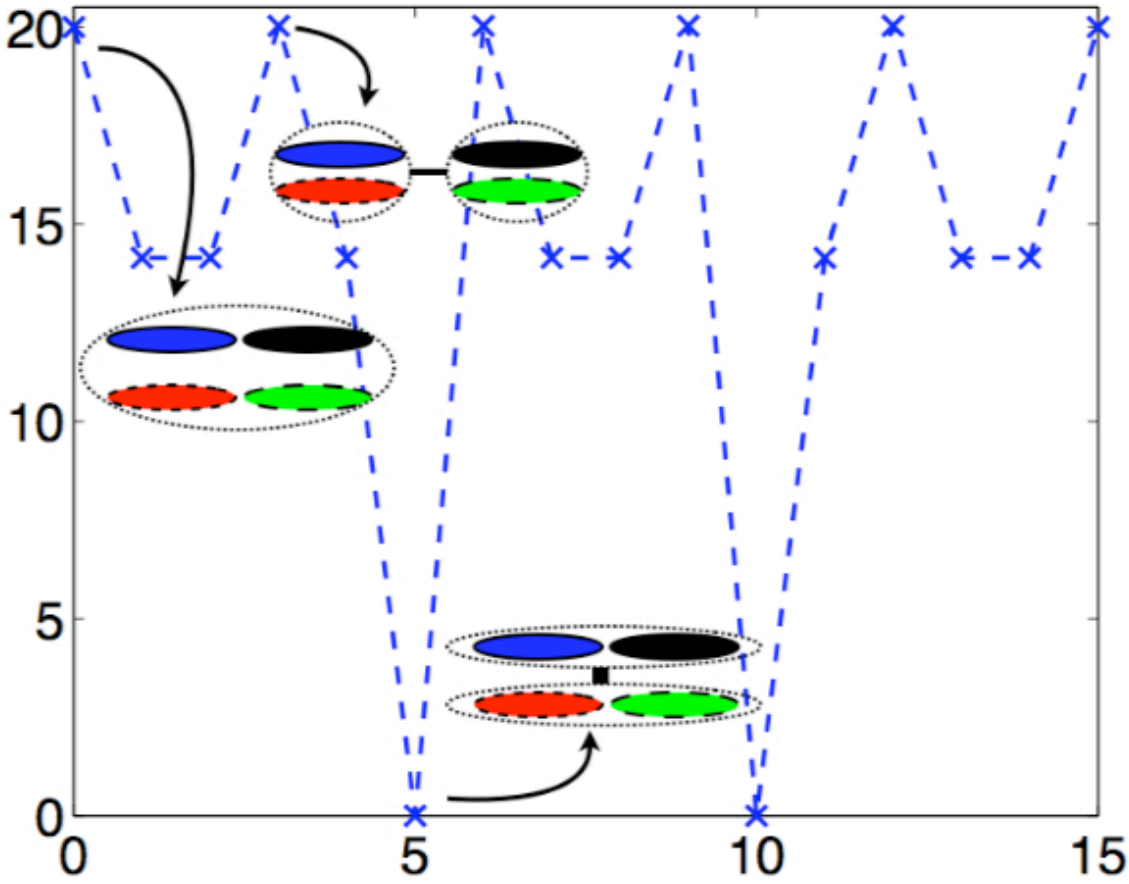

Figure 3.9. **Cost function for Kuramoto partition.** - For the OSN described in Fig. 3.7 where the individual oscillators are Kuramoto oscillators, where the frequency for nodes shown in blue and black are set to be -1, and those in red and green to be $+1$, with coupling strength $\sigma = 0.1$. If we assume that the nodes in the same color always belong to the same group, then the projection function P can be written by a row vector as $P = Q \otimes [1, 1, 1, 1, 1]$ where $Q = [q_1, q_2, q_3, q_4]$. If we further assume that the maximum number of groups is two, then all possible partitions can be represented by binary sequences of length 4 in Q. There are at most 16 of them (including degenerate cases). The cost function for each of them is shown (by blue cross) in the figure; curved arrows point to the partition associated with the function value in the figure where groups are indicated by dotted ellipses.

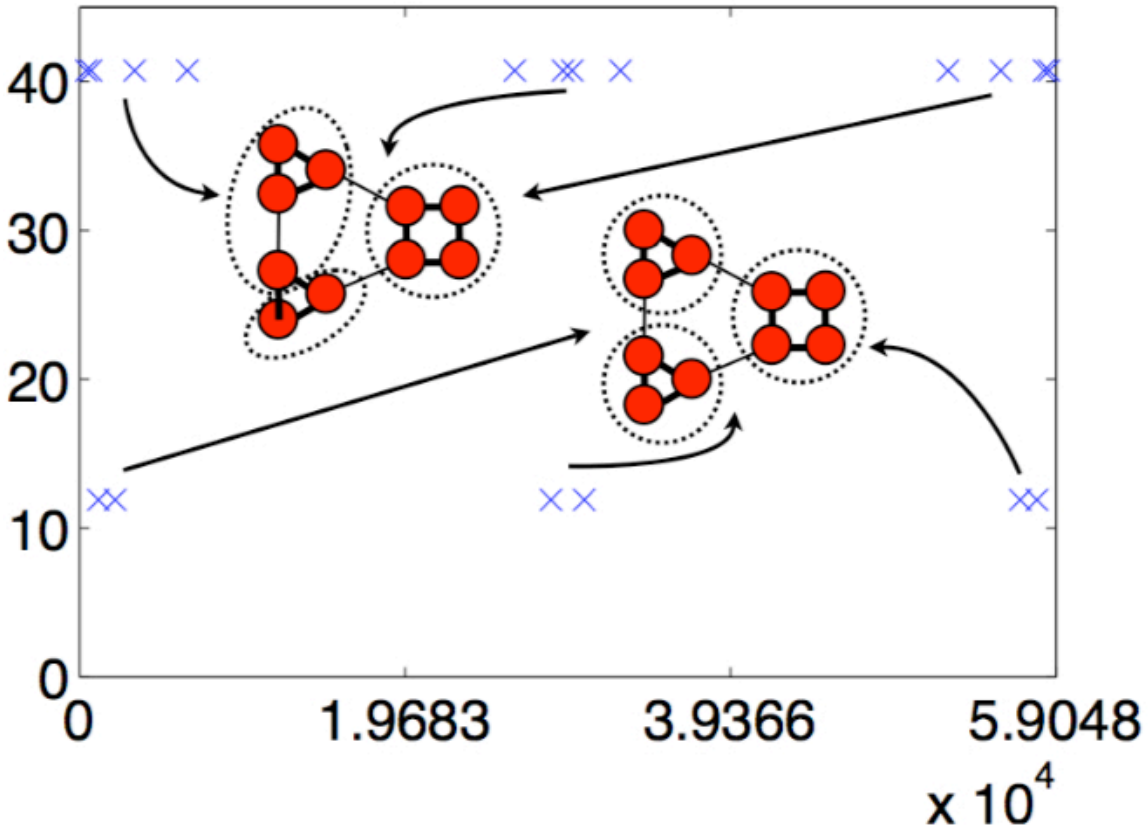

Figure 3.10. **Cost function for Kuramoto partition, another example.** - The notation is similar as what was described in Fig. 3.9, where here the example is an OSN described in Fig. 3.2.

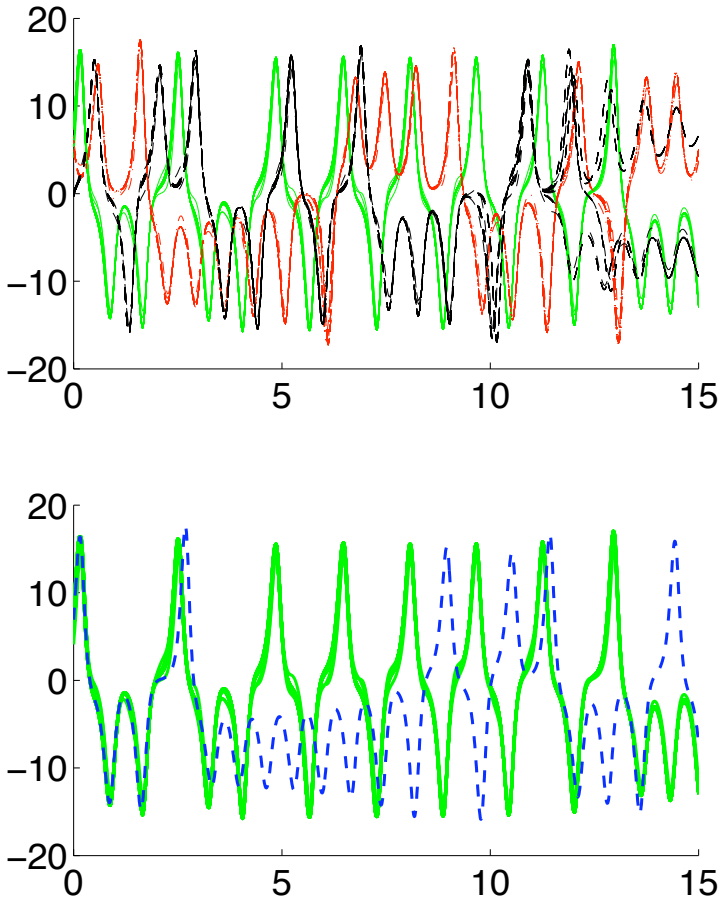

Figure 3.11. **Difficulty in judging the reduced order model for coupled chaotic oscillators.** - Here the network structure is the one shown in Fig. 3.3, but with identical Lorenz oscillators instead of Kuramoto oscillators, and the coupling function in Eq. 3.2 is simply $h(x) = x$, where the coupling strength $\sigma = 1$. In the upper panel we show the time series of the x components of all the oscillators. They again form three distinct clusters. Following the coarse-grain approach discussed in Section 2, we have a three-node coarse network. The picture shown in the lower panel shows the difference between time series for the x component from oscillators from one group of the original OSN (in green solid lines) and the one from its CGOSN (in blue dashed line). Different from the pictures we have seen for Kuramoto oscillator case, here the time series from CGOSN differs, after a short time period, from the ones from the original OSN. This might not be surprising after all, since all the oscillators are chaotic, and thus any tiny error in the model will lead to the divergence between time series from the model (CGOSN) to the original OSN. However, because of this, whether the CGOSN model is a good one in this case is in question.

Chapter 4

Modeling of Chaotic Oscillators: Preliminaries

In Chapter 4 we start with the problem of how to model a dynamical system from measurements (section 4.1); and review a classical concept called shadowing, and illustrate its relevance to the problem of modeling (section 4.2); then propose the concept of optimal shadowing, and show how it can be used to define a unique measure for model quality. The concepts introduced in this chapter will be used in Chapter 5 for the problem of multi-scale modeling in the case of chaotic oscillators.

4.1 Parameter Estimation from Measurements

4.1.1 Least Square Approach

In this section we briefly discuss the modeling of dynamical systems from empirical data. Given discrete measurements $\{x_t\}_{t=1}^{T}$, suppose that we know the form of the dynamics governing the evolution of x_t, described by a function $f : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}^m$, so that the input of f has the form (x, μ). Here x is the state variable, and μ is a parameter of the function f independent of x and does not change in time.

If the data were noiseless, i.e., $x_{t+1} \equiv f(x_t, \mu)$ for some parameter μ, then we would have:

$$\begin{bmatrix} f(x_1,\mu) \\ f(x_2,\mu) \\ . \\ . \\ . \\ f(x_{T-1},\mu) \end{bmatrix} = \begin{bmatrix} x_2 \\ x_3 \\ . \\ . \\ . \\ x_T \end{bmatrix}. \tag{4.1}$$

In the above matrix form equation, we are looking for a solution μ in terms of the data $\{x_t\}_{t=1}^T$. In general the equation may be nonlinear, and an exact solution is unlikely to be found. However, if the function f depends *linearly* on the parameter μ, i.e.,

$$f(x, \mu) = F(x)\mu \tag{4.2}$$

where $F(x) = [f_{ij}(x)]$ is an $m \times p$ matrix, and $\mu = [\mu_1, ..., \mu_p]^T$ is a $p \times 1$ vector representing the parameter, then a simple approximate solution can be easily obtained, as shown below.

In this case (i.e., where f has the form (4.2)) , Eq. (4.1) can be written as:

$$\begin{bmatrix} F(x_1) \\ F(x_2) \\ . \\ . \\ . \\ F(x_{T-1}) \end{bmatrix} \begin{bmatrix} \mu_1 \\ ... \\ \mu_p \end{bmatrix} = \begin{bmatrix} x_2 \\ x_3 \\ . \\ . \\ . \\ x_T \end{bmatrix}, \tag{4.3}$$

or simply in matrix form:

$$\boldsymbol{F}\mu = \boldsymbol{b} \tag{4.4}$$

where $\boldsymbol{F}$ (an $mT \times p$ matrix) and $\boldsymbol{b}$ (an $mT \times 1$ vector) represent the corresponding matrices in accordance with Eq. (4.3).

Eq. (4.4) is simply a set of linear equations, with unknown μ to be solved. However, in most interesting cases we have $mT \gg p$, and thus the system (4.4) is highly *overdetermined*. When noise is present in the system, i.e., $f(x_t, \mu) \neq x_{t+1}$, usually there is no solution that can satisfy Eq. (4.4) exactly. A classical approach in this

situation is the method of least squares, which can be used to find an approximate solution that minimizes the quantity:

$$||\boldsymbol{F}\mu - \boldsymbol{b}||_2 \tag{4.5}$$

where $||.||_2$ represents the Euclidean norm of a vector (Dem97).

Specifically, the *singular value decomposition* (SVD) (Dem97) of the matrix $\boldsymbol{F}$:

$$\boldsymbol{F} = U\Sigma V^T \tag{4.6}$$

can be used to construct a solution:

$$\mu = V\Sigma^{-1}U^T\boldsymbol{b} \tag{4.7}$$

which minimizes (4.5).

Note that the minimization of Eq. (4.5) is the same as minimizing the total squares of the *step-wise error* $||x_{t+1} - f(x_t, \mu)||_2$:

$$J(\mu) \equiv \sum_{t=1}^{T-1} ||x_{t+1} - f(x_t, \mu)||_2^2. \tag{4.8}$$

4.1.2 Example of a Quadratic Map

We illustrate the least square parameter estimation mentioned above by an example. Consider time series $\{x_t\}_{t=1}^{5000}$ generated by the discrete model:

$$x_{t+1} = 3.99x_t(1 - x_t) + \delta\eta_t \tag{4.9}$$

where η_t is a random noise term, distributed either uniformly on the interval $[-1, 1]$ (uniform noise) or according to a standard normal distribution; δ controls the magnitude of this noise. Suppose that all we have is the discrete measurements $\{x_t\}_{t=1}^{5000}$ and the form of the model

$$x_{t+1} = \mu x_t(1 - x_t), \tag{4.10}$$

where the goal would be to estimate μ.

Putting these into the form of Eq. (4.3), we can use Eq. (4.7) to solve for μ. Fig. 4.1 shows that for two different types of noise (either uniform or Gaussian as

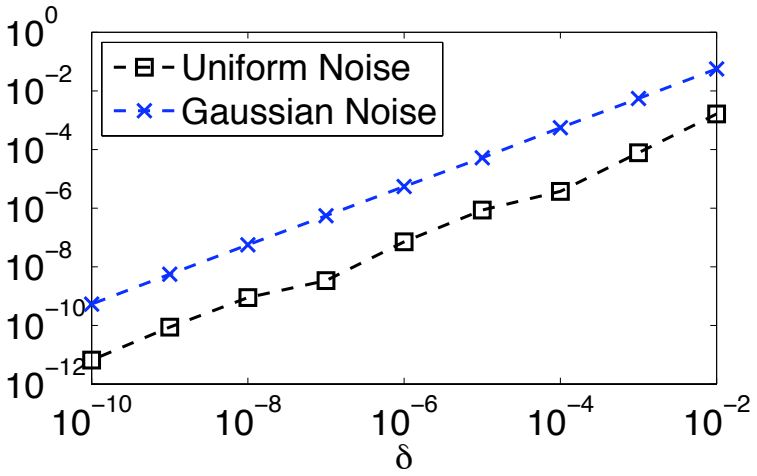

Figure 4.1. **Least square parameter estimation for quadratic maps: different types of noise.** - Time series $\{x_t\}_{t=1}^{5000}$ generated by the discrete model $x_{t+1} = 3.99x_t(1-x_t) + \delta\eta_t$ where η_t is a random noise term, distributed either uniformly on the interval $[-1, 1]$ (uniform noise) or according to a standard normal distribution; δ is a scalar, determining the level of noise. Shown is the difference between the estimated parameter using least square method and the 'true' parameter 3.99 in both cases (black squares for uniform noise case and blue crosses for Gaussian noise case).

described in the previous paragraph), the estimated parameter converges to the one used to generate the time series, 3.99, as the noise level δ goes to zero.

Note that if the underlying model has a more general form $f(x) = \mu_1 x - \mu_2 x^2$, similar methods can be used to estimate $\mu = [\mu_1, \mu_2]^T$. Fig. 4.2 shows the difference between μ_1, μ_2 and the actual parameters 3.99 used in the noisy model Eq. (4.9) where η_t is uniformly distributed in $[-1, 1]$. Both of the estimated μ_1 and μ_2 converge to 3.99 as noise approaches zero.

Although it does not seem too complicated for time series generated by discrete dynamics, the generalization to measurements from a continuous system (ODE or PDE) is not as simple. For analysis of the performance (convergence of estimated parameter to actual parameter relying on noise and grid size, etc.) of using least square like methods for such problems, the readers are referred to a recent paper by Yao and Bollt (YB07) and the references therein.

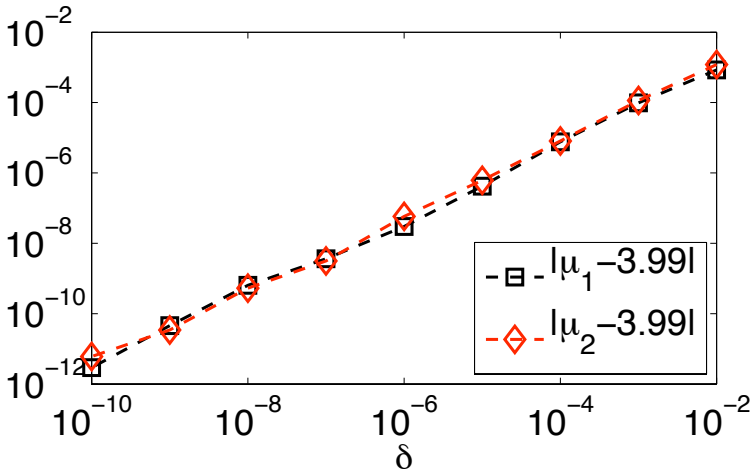

Figure 4.2. **Least square parameter estimation for quadratic maps: converge of two parameters.** - Parameter estimation for the model $x_{t+1} = \mu_1 x_t - \mu_2 {x_t}^2$ by least square method (4.7) from time series generated by $x_{t+1} = 3.99x_t - 3.99{x_t}^2 + \delta\eta_t$, where δ is the magnitude of the noise and η_t are uniformly distributed on $[-1, 1]$.

4.2 What is Shadowing?

4.2.1 Sensitive Dependence on Machine Precision: Is Chaos a Fiction?

Dynamical systems that possess chaotic behavior are known to have sensitive dependence on initial conditions. For example, consider the following simple chaotic map:

$$x_{t+1} = 2x_t \mod 1. \tag{4.11}$$

It is known that for *almost every* initial condition chosen from the interval $[0, 1]$, its orbit is chaotic, i.e., bounded, not asymptotically periodic, and has a positive Lyapunov exponent (equals 2). However, if one were to test this in *any* computer as long as the computer represents numbers in binary basis and its machine precision is *finite*, numerically there is simply no chaotic orbit. For example, in Matlab (version R2008a) (Mat), choose any initial condition from $[0, 1]$, and iterate according to Eq. (4.11), then after about 50 iterates, x_t becomes identically 0, i.e., the orbit converges to a boring periodic orbit $\{0, 0, ...\}$. See Fig. 4.3 as an illustration. Since the computer typically represents a number in a *finite* binary representation, the length of

the binary sequence of a number determines the number of steps it takes for the orbit of that number to converge to 0, since Eq. (4.11) simply shifts the binary sequence of a number one position at a time. Thus, one can indeed prove that for any binary computer, any orbit starting from initial conditions in $[0, 1]$ eventually converges to 0, and this convergence is usually surprisingly fast, as shown in Fig. 4.3.

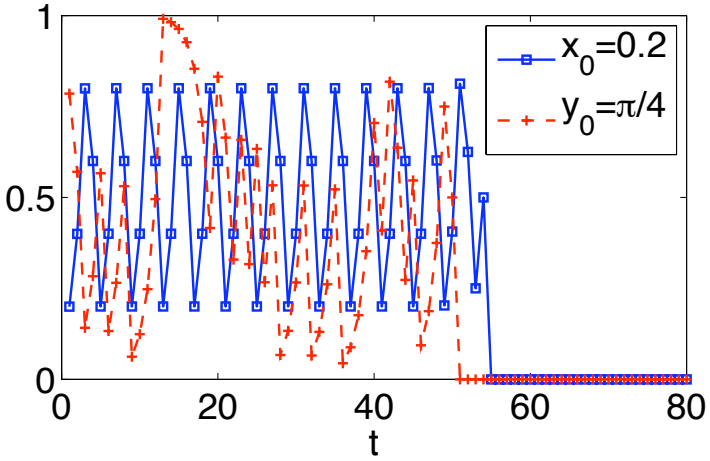

Figure 4.3. **Typical orbits of the** $2x \mod 1$ **map converge to zero (in a finite binary machine).** - Numerical computation of orbits from the discrete dynamics $x_{t+1} = 2x_t \mod 1$. Here the (blue) squares represent the orbit starting from $x_1 = 0.2$, and (red) "+" correspond to another orbit starting from $y_1 = \frac{\pi}{4}$, both orbits are computed in Matlab with double precision. After about 50 iterations, both trajectories converge to the orbit $\{0, 0, ..., 0, ...\}$.

Thus, the trajectories shown in Fig. 4.3 do not represent any true orbit from Eq. (4.11). It is obvious that starting from $x_1 = 0.2$, the trajectory should land precisely on a period-4 orbit $\{0.2, 0.4, 0.8, 0.6\}$; and perhaps less obviously, the trajectory starting from $y_1 = \frac{\pi}{4}$ should indeed be a chaotic orbit (in general, any orbit starting with an irrational number is a chaotic orbit), rather than becoming 0 after just several iterates.

This type of problem also occurs if we try to compute the same orbit with different numerical accuracy. Fig. 4.4 shows that, even if we use the same software and start with the same initial condition, the trajectories will typically diverge if they are numerically computed upon different machine precision.

The big question/puzzle arises now. If a computer is *never* capable of producing an orbit which comes exactly from a chaotic system, how can we trust anything that

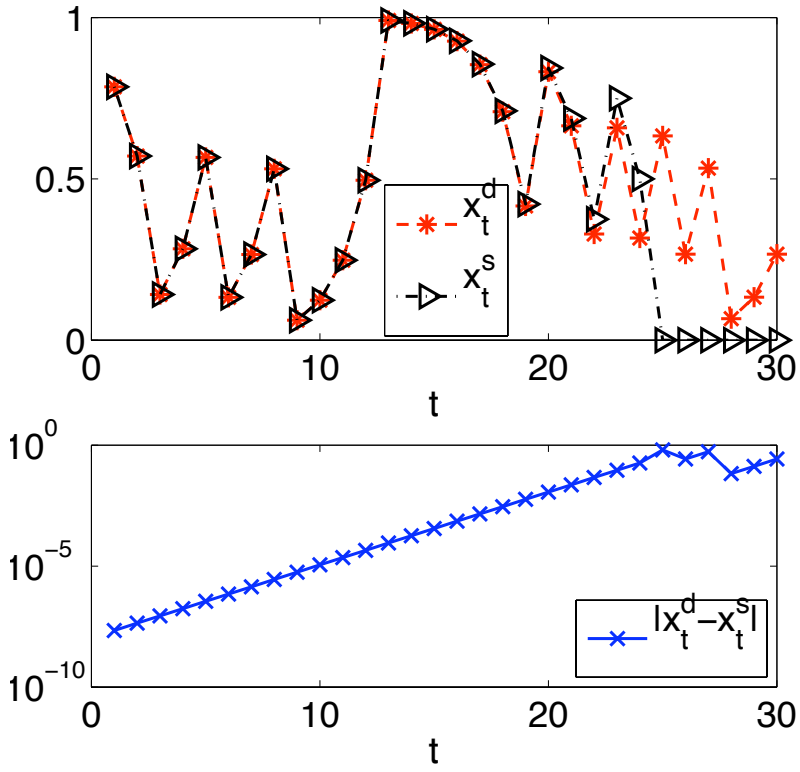

Figure 4.4. **Orbits start from the same initial condition computed with different machine precision.** - Numerically computed orbits from the discrete dynamics $x_{t+1} = 2x_t \mod 1$ starting from initial condition $\frac{\pi}{4}$ using either double precision, denoted by $\{x_t^d\}_{t=1}^{30}$ (plotted in red stars) or single precision ($\{x_t^s\}_{t=1}^{30}$, black triangles) arithmetic shown in the upper panel. In the lower panel we show the difference between the two numerical orbits in a log plot. The divergence of these two trajectories is exponential, as expected from the fact that the map $2x \mod 1$ is uniformly expanding.

comes out of compute simulations of chaotic systems, either a picture shown on the screen, or a number reported from calculations, such as Lyapunov exponents? If a chaotic system as simple as that described by Eq. (4.11) already suffers from such problems, how can we ever trust any 'chaotic like' patterns generated by computers for more complicated systems? Is chaos in reality simply a scientific fiction?

Starting from the next subsection, this whole section is devoted to the introduction of a concept called *shadowing* which addresses some of the above problems. The idea of shadowing was first introduced by Anosov (Ano67) and Bowen (Bow75) for uniformly hyperbolic systems (also known as *Axiom A diffeomorphisms*), and later on used by others (HYG87; HYG88; NY88; CKY88; GHYS90; SGY97; Sau02; Hay02) for more general chaotic systems that are not necessarily hyperbolic, but usually in dimensions no more than 2. This concept and massive amount of work in various aspects of the problem of shadowing has been nicely summarized in the books (Pil99) and (Pal00).

4.2.2 Infinite Shadowing for Hyperbolic Systems

As pointed out in the above subsection, for any given discrete dynamical system described in some form:

$$w_{t+1} = f(w_t); \tag{4.12}$$

In reality, *any* orbit $\{x_t\}_{t=1}^{T}$ generated according to Eq. (4.12) will not satisfy the equation exactly, but instead, follows:

$$x_{t+1} = f(x_t) + \eta_t, \tag{4.13}$$

where η_t represents the error made at computing x_{t+1} from x_t. For example, in a typical computer program, η_t represents the truncation error due to finite machine precision. In the following we shall assume that

$$\delta \equiv ||\eta||_\infty = \sup_t ||\eta_t||_2 \ll 1. \tag{4.14}$$

An orbit that satisfies Eq. (4.12) and Eq. (4.13) is usually called a δ pseudo orbit, emphasizing the fact that it is not exact, and the norm of the *step-wise error* η_t is uniformly bounded by the small positive number δ.

Our computers generate such pseudo orbits, for small δ. The orbits shown in Fig. 4.3 and Fig. 4.4 are indeed all δ pseudo orbits, although none of them seem to be interesting. We shall show an interesting pseudo orbit in Fig. 4.5, generated as follows: start with the initial condition $x_1 = \frac{\pi}{4}$; at each time step, compute a number $2x_t \pm \eta$ mod 1, where η is the Matlab machine precision; "$+$" and "$-$" are chosen with equal probability. This random process generates a pseudo orbit, shown in Fig. 4.5 (in red crosses); for comparison, the orbit starting with $\frac{\pi}{4}$ and computed by simply typing in $y_{t+1} = 2y_t \mod 1$ in Matlab is shown in black squares. The orbit $\{x_t\}$, although seems to be more noisy, looks more promising than the 'exact' one $\{y_t\}$ (which goes to 0 after a few iterates). Indeed one can check that even after a long number of steps the pseudo orbit generated as indicated here will still not be asymptotically periodic; it is very likely to be chaotic. One can play this trick for almost every initial condition chosen from the interval $[0, 1]$ and obtain a chaotic pseudo orbit. The question then is, how much can we rely on those chaotic like pseudo orbits?

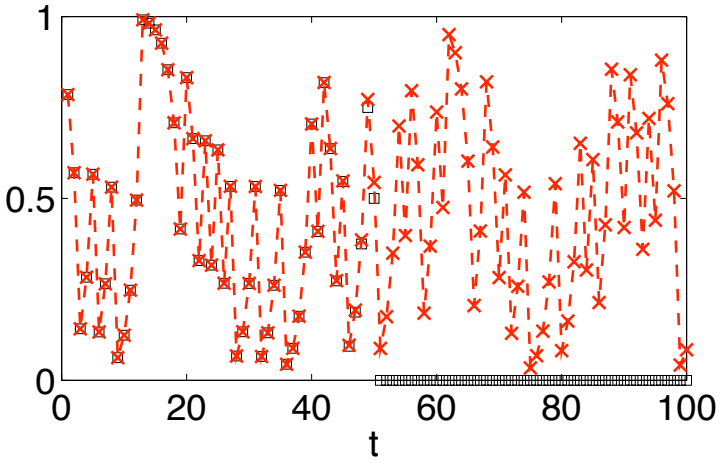

Figure 4.5. **Noisy** $2x \mod 1$ **map in Matlab.** - Shown in the figure is a noisy $2x$ mod 1 orbit generated as follows: start with initial condition $x_1 = \frac{\pi}{4}$, at each time step, compute a number $2x_t \pm \eta \mod 1$, where η is the Matlab machine precision; "$+$" and "$-$" are chosen with equal probability.

Given a δ pseudo orbit $x = \{x_t\}_{t=1}^T$, a true orbit $y = \{y_t\}_{t=1}^T$ is an orbit which satisfies

$$y_{t+1} = f(y_t). \tag{4.15}$$

That is, a true orbit is a pseudo orbit with $\delta = 0$. The true orbit y is said to ϵ *shadows*

the pseudo orbit x if the following holds:

$$||y_t - x_t||_2 < \epsilon, \ \forall t \in \{1, ..., T\}, \tag{4.16}$$

i.e., $||x - y||_\infty \equiv \sup_t ||x_t - y_t||_2 < \epsilon$. See Fig. 4.6 for illustration.

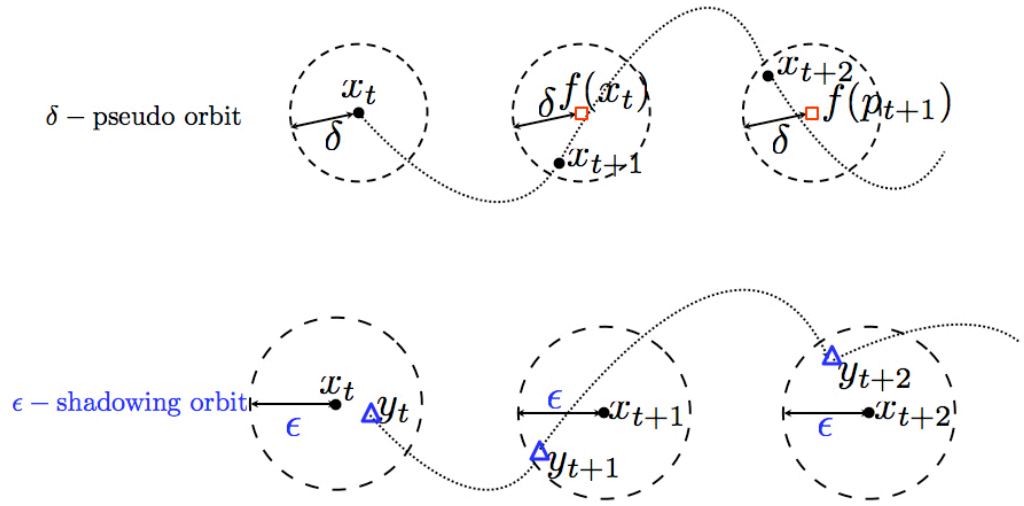

Figure 4.6. **Illustration of δ pseudo orbit and ϵ shadowing orbit.** - In the upper picture we illustrate the concept of a δ pseudo orbit, an orbit which suffers from finite step-wise-error $||x_{t+1} - f(x_t)||_2$ bounded by the number δ for all t. In the lower picture the idea of a shadowing orbit is illustrated. An ϵ shadowing orbit is a true orbit (or a 0 pseudo orbit) which ϵ shadows the pseudo orbit, i.e., $||x-y||_\infty \equiv \sup_t ||x_t - y_t||_2 < \epsilon$.

It can be shown that for f being the discrete dynamics (4.11), any δ pseudo orbit can be shadowed by a true orbit with ϵ no larger than 2δ. This is conveniently proved by using *symbolic dynamics*, and will be left as an exercise. [Hint: the dynamics (4.11) is *conjugate* to a *Bernoulli shift*, regarding numbers in $[0, 1]$ in their binary representation. A shadowing orbit can be found by feeding the errors made at each step to appropriate binary digits of the initial condition. For those of you who are not familiar with those concepts, (Jos05) can be a good reference.]

Thus, although the chaotic-like orbit shown in Fig. 4.6 is not a true orbit starting from the initial condition as we specified, there exists a true orbit that lies close to (or by technical language, shadows) it, *step by step*. Since the resolution of our screen probably is not high enough to differ between the two orbits, it is safe in this case to say that what we observe is a chaotic orbit.

In general, the existence of a shadowing orbit for a pseudo orbit with infinite

length (i.e., $T = \infty$) is guaranteed when the pseudo orbit is generated from a *uniformly hyperbolic system* with small enough step-wise-error. The original form of this shadowing result was reported in (Bow75). Here we review a modern variant of it which is based on the contraction mapping theorem for bounded linear operators on Banach spaces (Pal00).

Lemma 4.1 (Infinite Shadowing (Pal00)) *Let S be a compact hyperbolic set for a C^1 diffeomorphism $f : U \to \mathbb{R}^m$. Then there exist positive constants δ_0, σ_0 and M depending only on f and S such that if $g : U \to \mathbb{R}^m$ is a C^1 diffeomorphism satisfying*

$$||f(x) - g(x)|| + ||Df(x) - Dg(x)|| \leq \sigma \quad \text{for } x \in U \tag{4.17}$$

with $\sigma \leq \sigma_0$, any δ pseudo orbit of f in S with $\delta \leq \delta_0$ is $\epsilon-$shadowed by a unique true orbit of g with $\epsilon = M(\delta + \sigma)$.

Definition 4.1 (Hypberbolic Set (Pal00)) *A compact set $S \subset U$ is said to be hyperbolic if*

1. *S is invariant, that is, $f(S) = S$;*

2. *there is a continuous splitting $\mathbb{R}^n = E^s(x) \oplus E^u(x)$, $x \in S$ such that the subspaces $E^s(x)$ and $E^u(x)$ have constant dimensions; moreover, these subspaces have the invariance properties*

$$Df(x)(E^s(x)) = E^s(f(x)), \; Df(x)(E^u(x)) = E^u(f(x)) \tag{4.18}$$

and there are positive constants K_1, K_2 and $\lambda_1 < 1, \lambda_2 < 1$ such that for $k \geq 0$ and $x \in S$

$$Df^k(x)\xi \leq K_1\lambda_1^k||\xi|| \text{ for } \xi \in E^s(x) \tag{4.19}$$

and

$$Df^{-k}(x)\xi \leq K_2\lambda_2^k||\xi|| \text{ for } \xi \in E^u(x). \tag{4.20}$$

4.2.3 Finite Shadowing for Non Hyperbolic Systems

Although it becomes evident that infinite shadowing exists for uniformly hyperbolic systems, it took a while for mathematicians to realize that those systems are in some sense not generic. Indeed, the opposite might as well be true: most interesting chaotic systems seem to be non hyperbolic. For non hyperbolic systems, the proof used in (Ano67; Bow75; Pil99; Pal00) does not work anymore, and there is currently no general result about the existence of infinite shadowing (it is likely that there simply is no infinite shadowing in general for such systems, as observed in many examples reported in the literature (Pil99; Pal00)). Instead, the best one can do is to check, for a given, *finite* pseudo orbit, whether a shadowing orbit exists.

Lemma 4.2 (Finite Shadowing (Pal00)) *Let $f : \mathbb{R}^m \to \mathbb{R}^m$ be a C^2 map and $\{x(t)\}_{t=1}^T$ be a δ pseudo orbit. For a given right inverse L^{-1} of the linear operator $L : (\mathbb{R}^m)^T \to (\mathbb{R}^m)^{T-1}$, defined for $u = \{u(t)\}_{t=1}^T \in (\mathbb{R}^m)^T$ by*

$$(Lu)_t = u_{t+1} - Df(x_t)u_t \quad \text{for } t = 1, ..., T-1, \tag{4.21}$$

set

$$\epsilon = 2||L^{-1}||\delta, \tag{4.22}$$

where the norm of L^{-1} is the operator norm with respect to the supremum norm on $(\mathbb{R}^m)^T$ and $(\mathbb{R}^m)^{T-1}$. Next let

$$M = \sup\{||D^2 f(v)|| : v \in \mathbb{R}^m, ||v - x_t|| \leq \epsilon \text{ for some } t = 1, ..., T.\} \tag{4.23}$$

Then if

$$2M||L^{-1}||^2\delta \leq 1, \tag{4.24}$$

the δ pseudo orbit $\{x(t)\}_{t=1}^T$ of f is $\epsilon-$shadowed by a true orbit of f.

Note that this lemma provides a sufficient, but not necessary condition on shadowing. In practice, if one is given a pseudo orbit with considerable step-wise error δ, it is likely that this lemma does not hold directly, while there still exists a shadowing orbit. A practical (but not yet rigorous) approach to get around it is through

pseudo shadowing, a concept to be discussed in the following, or look at Fig. 4.7 as an example.

4.3 Optimal Shadowing

4.3.1 Pseudo Shadowing: A Rescue for Imperfect Computers

Some important work regarding shadowing, such as (FS91) relies on using modern computers to compute, for a given pseudo orbit, a shadowing orbit. For example, given a δ pseudo orbit, the method in (FS91) is intended to find a true orbit which minimizes the l_2 distance from the pseudo orbit. Specifically, gradient descent types of method were suggested for numerically finding an approximate solution to the minimization problem (RJ02; Jud08). However, as we mentioned many times before, our computers usually have *finite* machine precision. A numerically computed "shadowing" orbit is unlikely to be a true orbit, but may very well be another pseudo orbit with a smaller δ.

The concept of *pseudo shadowing* is meant to resolve this problem, a similar idea was also mentioned in a recent work by Judd (Jud08). Recall that for a given discrete dynamical system $f : \mathbb{R}^m \to \mathbb{R}^m$, a sequence of numbers $\{x_t\}_{t=1}^T$ is called a δ pseudo orbit if for any $t \in \{1, ..., T-1\}$, $||x_{t+1} - f(x_t)||_2 < \delta$. A true orbit $\{y_t\}_{t=1}^T$ is an orbit which satisfies: $||y_{t+1} - f(y_t)||_2 = 0$ for each t. The true orbit is said to ϵ shadows the pseudo orbit $\{x_t\}_{t=1}^T$ if,

$$||y - x||_\infty \equiv \sup_t ||y_t - x_t||_2 < \epsilon. \tag{4.25}$$

Now suppose that we are given a δ pseudo orbit $x = \{x_t\}_{t=1}^T$, while the finite shadowing lemma does not hold directly; instead, we are only able to find another $\tilde{\delta}$ pseudo orbit $\{y_t\}_{t=1}^T$ with $\tilde{\delta} < \delta$, and such that

$$||y - x||_\infty < \tilde{\epsilon}. \tag{4.26}$$

For this $\tilde{\delta}$ pseudo orbit $\{y_t\}_{t=1}^T$, it is possible that the finite shadowing lemma holds, with ϵ being the shadowing distance from a true orbit $\{z_t\}_{t=1}^T$ (exists but not computed

explicitly) to this $\tilde{\delta}$ pseudo orbit, so that

$$||z - y||_\infty < \epsilon. \tag{4.27}$$

Then we may conclude, by the triangle inequality

$$||z - x||_\infty \leq ||z - y||_\infty + ||y - x||_\infty < \epsilon + \tilde{\epsilon}, \tag{4.28}$$

that the original δ pseudo orbit $\{x_t\}_{t=1}^T$ can be shadowed by a true orbit with shadowing distance no larger than $\epsilon + \tilde{\epsilon}$.

The $\tilde{\delta}$ pseudo orbit $\{y_t\}_{t=1}^T$ plays an important role in the above process of proving shadowability of general pseudo orbits, and will be referred to as a *pseudo shadowing orbit* of the original pseudo orbit.

As an example, Fig 4.7 shows how direct use of the finite shadowing lemma might be misleading, and how the idea of pseudo shadowing might be useful in practice for showing the existence of a shadowing orbit for given noisy orbits.

4.3.2 Optimal Shadowing: Theorems and Algorithms for $1D$ Maps

Since shadowing can be applied to any noisy orbit regardless of the source of noise, it becomes interesting to ask, what is the best possible shadowing orbit for a given observed orbit? For example, given $x = \{x_t\}_{t=1}^T$ and assume that this data comes from a discrete dynamical system: $f : \mathbb{R}^m \to \mathbb{R}^m$. One may find a shadowing orbit with shadowing distance ϵ not necessarily impressive. For example, for any sequence of numbers $x = \{x_t\}_{t=1}^T$ where each $x_t \in [0, 1]$, even if $T = \infty$, for any $f : [0, 1] \to [0, 1]$, there is a true orbit $y = \{y_t\}_{t=1}^T$ of f which shadows x, with shadowing distance no larger than $\epsilon = 1$ (Trivial!). It only becomes interesting if the shadowing orbit shadows the pseudo orbit with a small enough ϵ. But how small can this ϵ be for general systems? When we do find a shadowing orbit, how can we be sure that it is not a trivial one? What is the minimal possible ϵ for a given pseudo orbit? Such questions are important in bounding the error of using noisy orbits to compute statistics for certain dynamical systems, and for measuring model quality, a topic to be discussed in the next section.

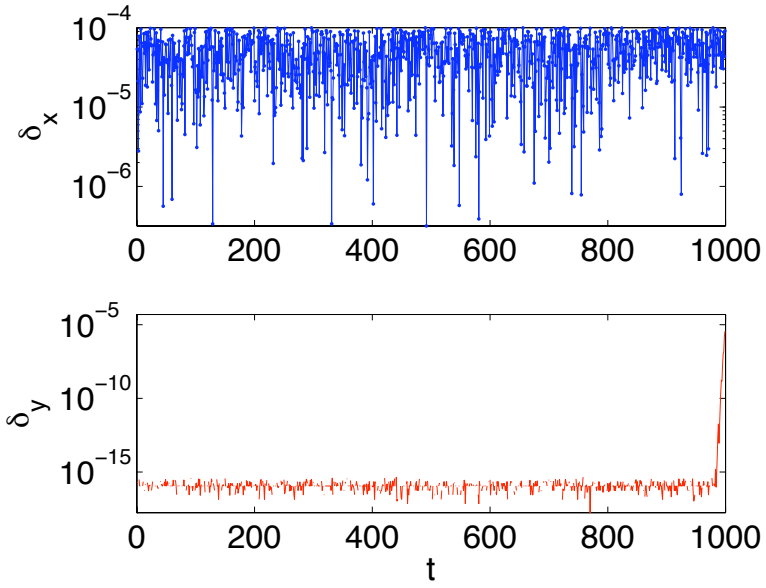

Figure 4.7. **Illustration of pseudo shadowing.** - In this example a pseudo orbit $x = \{x_t\}_{t=1}^T$ is generated by starting with $x_1 = 0.9501$ and following the rule: $x_{t+1} = 3.9999x_t(1 - x_t) + \eta_t$, where η_t are independent random noise distributed uniformly on the interval $[-10^{-4}, 10^{-4}]$. The orbit x is a pseudo orbit with $\delta_x \approx 10^{-4}$; using the equations (4.21), (4.22), (4.23), and (4.24), it can be approximated that $\epsilon_x = 2||L_x^{-1}||\delta_x \approx 0.0245$, and $2M_x||L_x^{-1}||^2\delta_x \approx 23.9624 > 1$, so the shadowing lemma does not apply. However, we are able to find another pseudo orbit $y = \{y_t\}_{t=1}^T$ (details of computing such an orbit are omitted), with $\delta_y \approx 3.6561 \times 10^{-6}$, and $||x - y||_\infty \approx 0.0027$. Now for this y orbit, we have: $\epsilon_y = 2||L_y^{-1}||\delta_y \approx 4.9315 \times 10^{-4}$, and $2M_y||L_y^{-1}||^2\delta_y \approx 0.2661 < 1$. Thus the orbit y can be shadowed by a true orbit, with $\epsilon_y < 5 \times 10^{-4}$, and by the triangle inequality of pseudo shadowing, we can conclude that the original orbit x can also be shadowed by some true orbit, with shadowing distance smaller than $\epsilon_x + \epsilon_y \approx 0.0032$.

In this subsection we introduce this problem of optimal shadowing, and show some theoretical and computational results for $1D$ discrete dynamical systems. Generalizations to higher dimensions and to continuous systems are possible, but will not be discussed in this thesis.

Given an orbit $x = \{x_t\}_{t=1}^T$, and a dynamical system $f : \mathbb{R}^m \to \mathbb{R}^m$, a true orbit $y = \{y_t\}_{t=1}^T$ (recall that a true orbit is one such that $y_{t+1} = f(y_t)$ for all t) is called an *optimal shadowing orbit* if for any other true orbit $z = \{z_t\}_{t=1}^T$,

$$||z - x||_\infty \geq ||y - x||_\infty. \tag{4.29}$$

The shadowing distance $||y-x||_\infty$ from an optimal shadowing orbit y to the observed pseudo orbit x is referred to as the *optimal shadowing distance*, denoted by $\epsilon_{opt} \equiv ||y - x||_\infty$.

In the following we assume that f is a continuous $1D$ map, i.e., $f \in C^0(S)$ where S is a compact subset of $\mathbb{R}$. Given a pseudo orbit $x = \{x_t\}_{t=1}^T$, define, for $\epsilon > 0$, the following sequence of intervals $\{I_t\}_{t=1}^T$:

1. $I_1 = [x_1 - \epsilon, x_1 + \epsilon]$
2. $I_{t+1} = f(I_t) \cap [x_{t+1} - \epsilon, x_{t+1} + \epsilon]$ for $t = 1, 2, ..., T - 1$.

Then the pseudo orbit x can be shadowed by a true orbit from f with shadowing distance no larger than ϵ if and only if $I_T \neq \emptyset$. [Why? Hint: the true orbit has to ϵ shadows p, so it is necessary to constrain each interval in the way above. The existence of a shadowing orbit is obvious when the condition holds.] The above criteria will be referred to as the *forward iteration criteria*.

Note that one may also use the following *backward iteration criteria*, based on f^{-1} (preimage of f). Define

1. $J_T = [x_T - \epsilon, x_T + \epsilon]$
2. $J_t = f^{-1}(J_{t+1}) \cap [x_t - \epsilon, x_t + \epsilon]$ for $t = 1, 2, ..., T - 1$.

In this case, the pseudo orbit x is ϵ shadowable if and only if $J_1 \neq \emptyset$.

To find a bound for the optimal shadowing distance, we simply need a decreasing sequence of numbers $\{\epsilon_k\}$, and check the forward (or backward) shadowing criteria until it fails for some ϵ_K. Then the optimal shadowing distance is simply bounded by: $\epsilon_K < \epsilon_{opt} \leq \epsilon_{K-1}$. In this thesis, we always choose this decreasing sequence of numbers $\{\epsilon_k\}$ in such a way that $\epsilon_{k-1} - \epsilon_k < 10^{-15}$. In cases where $\epsilon_{K-1} \gg 10^{-15}$, we can simply approximate ϵ_{opt} by ϵ_{K-1} without having to actually compute ϵ_{opt}.

In theory, the above two criteria are equivalent. Fig. 4.8 shows an example of computing the optimal shadowing distance using both the forward and backward iteration for a given pseudo orbit and a family of quadratic maps. In this example the two methods give approximately the same results. However, in general there is no guarantee that the two methods will coincide, in practice, since numerical computation might cause error and the type of error is usually different for forward and backward iterations. One needs to be careful about which one to use (or maybe use a mixture of both), depending on the nature of the dynamics of f. We have not been able to develop rigorous results about this criteria at this time[1], and in this thesis we will use the forward iteration method without going into the details about the computation.

We would like to comment here that the forward iteration approach is actually a simplified version of the *containment* process proposed originally in (GHYS90) by Grebogi et al., for general nonhyperbolic systems. However, in their paper the authors were interested in establishing lower bounds for the shadowing distance, thus terminating the process whenever $f(I_t) \supset [p_{t+1} - \epsilon, p_{t+1} + \epsilon]$ fails. We notice here that the failure of satisfying this criteria does not necessarily imply a *glitch point* (defined for given ϵ as the point where there is no shadowing orbit which is able to shadow further).

[1]One possible solution is through *interval arithmetic* as described in (HYG87), which we have not explored in enough depth in order to carry out a general theory about optimal shadowing.

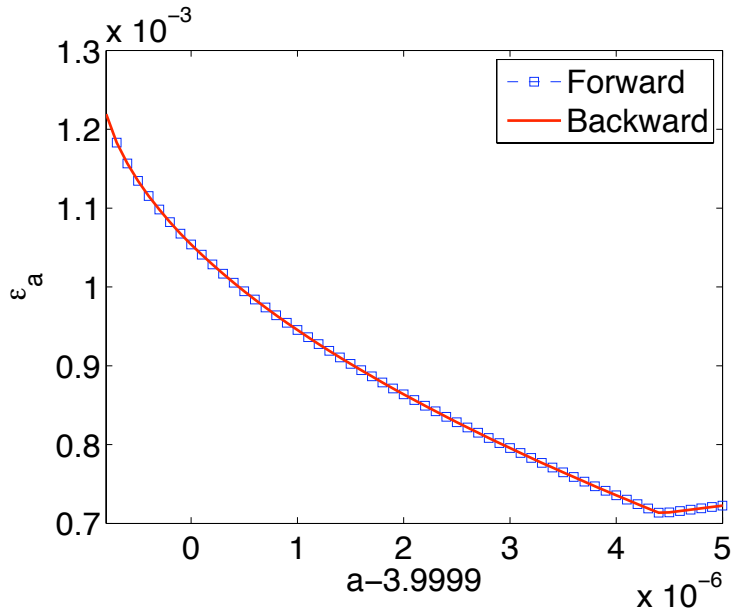

Figure 4.8. **Comparison of the forward and backward method in finding shadowing orbits.** - Here a pseudo orbit $x = \{x_t\}_{t=1}^{1000}$ is generated by $x_{t+1} = 3.9999x_t + 10^{-5}\eta_t$ where η_t are independent random variables distributed uniformly on $[-1, 1]$. In the picture we show shadowing distance from an optimal shadowing orbit from dynamics $y_{t+1} = ax_t(1 - xt)$ with different parameters a. Results using both the forward (blue squares) and backward iteration (red curve) are shown. These two methods seem to coincide at least of a small neighborhood around the "true" parameter 3.9999.

4.4 Measuring Quality of Modeling via Shadowing

Given an observed time series $\{x_t\}_{t=1}^T$, and a family of parameter-dependent models (with form f and parameter μ)

$$\begin{aligned} f : \mathbb{R}^m \times \mathbb{R}^p &\to \mathbb{R}^m \\ (x, \mu) &\mapsto f(x, \mu), \end{aligned} \tag{4.30}$$

section 3.1 describes how the Euclidean norm can be used to compare different model qualities for different choices of μ, and the least square method can be used to select the optimal model in the sense of minimizing the total square of step-wise errors. However, as we see in the previous two sections, for dynamical systems that are chaotic, it is often more interesting and important to look at the sup norm instead of the Euclidean norm, that is, we prefer a model which is capable of producing an orbit that follows the observed time series *as close as possible* and *as long as possible*. Since a model is supposed to generate only true orbits, and the empirical time series often suffers from noise from step to step, the above problem can naturally be recast as an optimal shadowing problem. In this section we define model quality in terms of its ability to shadow observed time series. Specifically, shadowing distance and shadowing time may both be used as a measure for the model quality, which usually provides different conclusions as opposed to the least square method.

4.4.1 Shadowing Distance vs. Shadowing Time

Shadowing Distance ——

For a given pseudo orbit (i.e., a noisy time series) $x = \{x_t\}_{t=1}^T$ and a candidate model f. The model quality of f may be measured by the optimal shadowing distance using

$$Q_\epsilon(f) = \epsilon_{opt}(f|x). \tag{4.31}$$

In Fig. 4.9 we show how the optimal shadowing distance may be used to judge model quality. For comparison, results based on minimizing step-wise error criteria (i.e., method of least squares) are also shown in the same figure. In general the two measure will not be the same, although they might give close conclusions in some

situations.

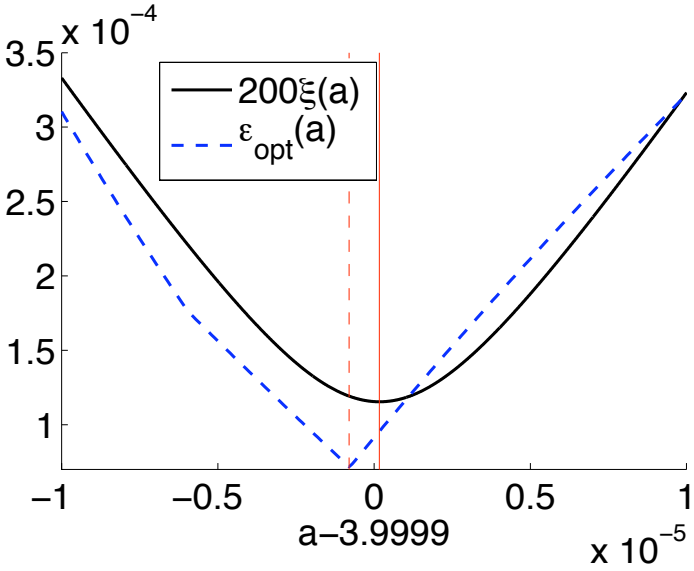

Figure 4.9. **Comparison of parameter estimation based on least square criteria and shadowing criteria.** - A pseudo orbit is generated by the process described in the caption of Fig. 4.8, with the only difference that the noise level is set to 10^{-6} instead of 10^{-5}. In this figure the black quadratic-like curve shows, for each a, the value $\xi(a) \equiv \sqrt{\left(\sum_{t=1}^{T}\left[x_{t+1} - f(x_t, a)\right]^2\right)/T}$ magnified by a factor of 200, while the blue dashed curve shows the optimal shadowing distance for each model $f(x, a)$. The two vertical lines are used to indicate the location of the minimal points of the two curves. Notice a slight difference between the location of the minimal points, as a result of the difference between the two model quality criteria.

Shadowing Time ——

A δ pseudo orbit may not in general be shadowed by a true orbit for every $\epsilon > 0$. Alternatively, one may ask, for given ϵ, what is the largest T^* such that the finite sub-orbit $\{x_t\}_{t=1}^{T^*}$ can be ϵ shadowed by f, while $\{x_t\}_{t=1}^{T^*+1}$ cannot? Note that it is important to ask for the largest T^*, since otherwise it is easy to find a shadowing orbit that can only shadow a few initial points of the orbit x. Using optimal shadowing time, the quality of a model may be assessed via:

$$Q_T(f) = T^*(f|p, \epsilon). \tag{4.32}$$

This criteria has been similarly proposed (but without concern that one should look for the largest, instead of an arbitrary T^*) for nonlinear model quality evaluation specifically motivated by weather forecasting (Gil98).

We illustrate this also by an example, as shown in Fig. 4.10.

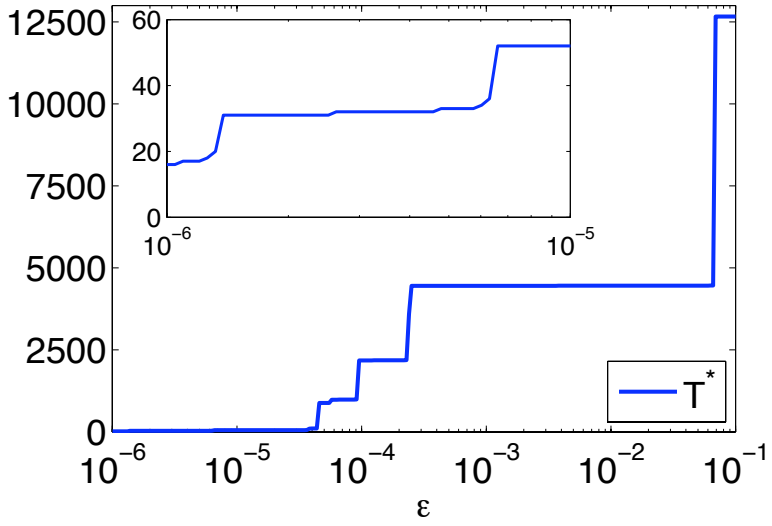

Figure 4.10. **Judging model quality via optimal shadowing time.** - In this example we generate a long pseudo orbit similar to the process described in Fig. 4.8, but with noise level 10^{-6}, and length $T = 50000$. For this pseudo orbit, we compute, for the model $f(x) = 3.9999x(1-x)$, approximately the optimal shadowing time T^* for different levels of shadowing distance ϵ. The optimal shadowing time is computed by applying the forward iteration until $I_{T^*+1} = \emptyset$, and this T^* is used as an approximation to the actual optimal shadowing time, for the given ϵ. The inset shows the part of this curve from $\epsilon = 10^{-6}$ to $\epsilon = 10^{-5}$. The stair shape of the curve seems to be interesting, and motivates further exploration for an explanation.

4.4.2 Ensemble Average Criteria

When a complicated system generates time series, we may not be interested in a single, infinite trajectory, but instead, different trajectories with finite length. For example, if our goal is to model the weather on a monthly basis, we may prefer to judge the model quality by looking at how many times, on different occasions, the model successfully describes the actual weather pattern, instead of judging the model quality by its behavior during the whole time.

Fig. 4.11 is an example showing how the ensemble average criteria may be used as a way to judge model quality when different pseudo orbits are considered.

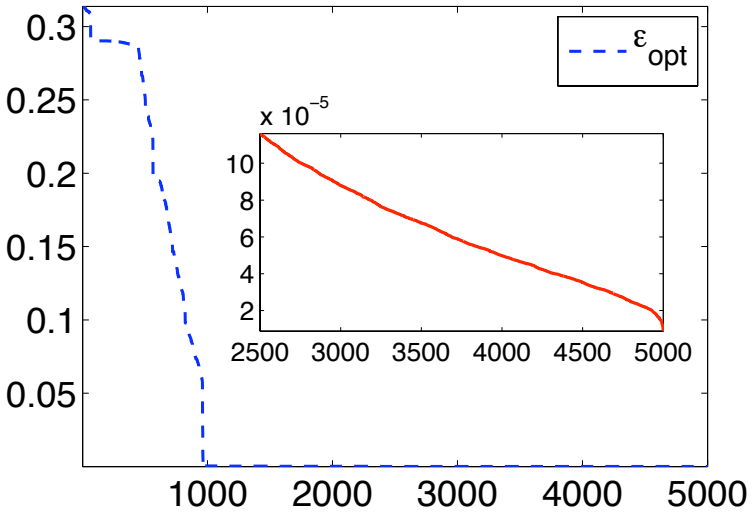

Figure 4.11. **Shadowing of an ensemble of pseudo orbits.** - In this example we generate an ensemble of 5000 different pseudo orbits by the random process described in Fig. 4.8, but with noise level 10^{-6}. For each of the pseudo orbits, we compute, for the model $f(x) = 3.9999x(1-x)$, the optimal shadowing distance using the forward iteration method. These 5000 ϵ_{opt} are then sorted in decreasing order and shown in the figure; inset is the enlarged part from index 2500 to 5000, showing that for about half of the orbits the optimal shadowing distance is below 10^{-4}. The average ϵ_{opt} for this example is about 0.0426, and the standard deviation is about 0.0942.

Chapter 5

Model Reduction of Coupled Chaotic Oscillators: A Shadowing Approach for Judging Model Quality

5.1 A Shadowing Approach for Measuring Model Quality for Chaotic Systems

5.1.1 Difficulty in Judging a Chaotic Model

A common method to compare models is to measure the difference between the function representing the model and some empirical fitting of data, under suitable norms such as the l_2 norm because of its convenience. However, for chaotic systems, such criteria to compare the two functions for determining whether a model is good or not may not be as useful. Fig. 5.1 shows that, two functions can have small distance in the functional space, while exhibit dramatically different dynamical properties.

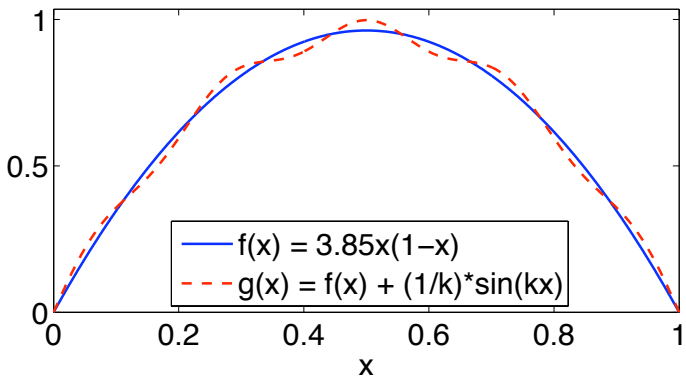

Figure 5.1. **Comparing models directly in the function space.** - Illustration of two functions close in the l_2 sense but not as close as two dynamical systems. Assume that the two systems are governed by the discrete dynamics $x_{n+1} = f(x_n)$ and $x_{n+1} = g(x_n)$, where $f(x) = 3.85x(1-x)$ (blue solid line) and $g(x) = f(x) + \frac{1}{k}\sin(kx)$ (red dashed line). Shown in the figure is for $k = 9\pi$. In general, $||f - g||_2 \equiv (\int_0^1 ||f(x) - g(x)||^2 dx)^{\frac{1}{2}} \leq \frac{1}{k}$. The distance can be made as small as possible, by tuning k large; however, complexity of the dynamics f, measured by topological entropy, can be increased arbitrarily by choosing k large.

Another common way to judge model quality is by comparing trajectories, which can easily be misleading for chaotic systems because of the sensitivity to initial conditions. When random noise or modeling error is introduced, as is always the case in any practical situations, even a seemingly perfect model may suffer from conflicting judgements. Fig. 5.2 shows that, if one does not exercise enough care, he may come to conflicting conclusions: on one hand, the model is in some sense *perfect*, since its form matches exactly what is driven the state variable in the absence of noise; on the other hand, the model seems to be *bad*, because by comparing the orbit generated by the model, starting with the same initial point as the observed time series and the observed orbit, there is considerable difference between the two orbits, due to the sensitive dependence on initial conditions, a typical property of chaotic systems. Indeed, the problem of matching numerical orbit and model generated orbit for chaotic systems lies in the area called *shadowing*, a classical concept in dynamical systems reviewed in the previous chapter.

5.1.2 Judging a Chaotic Model via Shadowing

Instead of judging a model by simply computing the distance between some choice of model-generated orbit and the observed orbit, which usually leads to non-definite answers, we ask, what is the *best* orbit (referred to as an *optimal shadowing orbit*) the model can produce, to match the observed orbit? Given an observed orbit $\{p_t\}_{t=1}^T$, and a candidate model $g : \Re^m \to \Re^m$ which describes a discrete dynamical system, the quality of the model g for the observation is measured by the *optimal shadowing error*

$$\epsilon_{opt} \equiv \inf_{x_1 \in \Re^m} ||x - p||, \tag{5.1}$$

where $p = \{p_t\}_{t=1}^T$ is the observed orbit, and $x = \{x_t\}_{t=1}^T$ satisfying $x_{t+1} = g(x_t)$ is a model generated orbit starting at initial condition x_1, and $||.||$ is usually taken as the sup norm (or max norm for finite T). For deterministic systems, this question is equivalent to finding the initial point which leads to a true orbit which can match the noisy orbit best, measured in some norms.

5.2 Judging Quality of a Model Reduction

Thus shadowing can further be used for the purpose of evaluating the model quality of a reduced order model of high dimensional chaotic time series. This approach allows us to quantify the quality of a model in an appropriate way, and offers a possibility for comparing different possible reduced order models, which is not likely to be achieved by traditional methods.

In the case of judging model reduction, since the high dimensional system and its reduced order model necessarily generate time series of different dimensions, there is currently no natural, canonical way of matching those models. Our approach to solve this problem can be described by the diagram shown in Fig. 5.3. Given a high dimensional system and its candidate reduced order model, instead of direct comparison, we propose the following procedure to measure the model quality: first, we generate time series from the original system, this time series is of course of high dimensions; next, dimensionality reduction techniques will be used, to extract a low dimensional representation of the time series (if there exist one); finally, shadowing

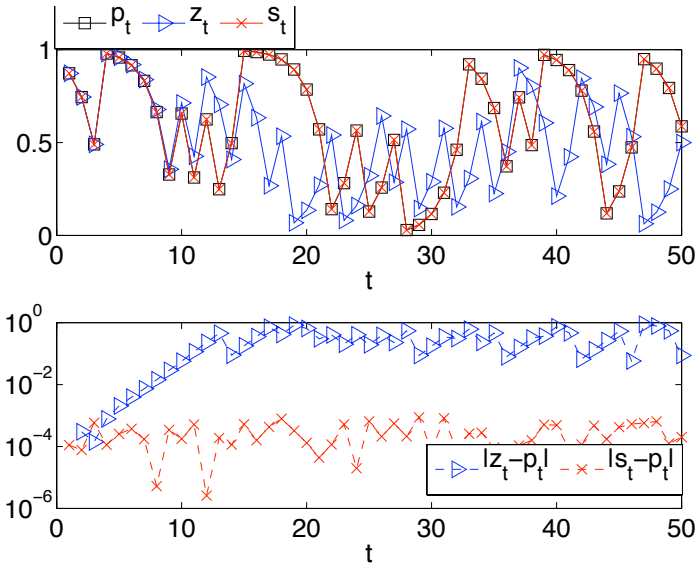

Figure 5.2. **Illustration of the difficulty in judging a model by comparing orbits directly.** - In the upper panel, a noisy numerical orbit $\{p_t\}_{t=1}^{50}$ of the logistic map $x_{t+1} = 4x_t(1 - x_t)$ is shown (in black squares). This orbit satisfies $p_{t+1} = 4p_t(1 - p_t) + \delta_t$ where δ_t is uniformly distributed, and is of the level 2^{-10}. The (blue) triangles are a true, noiseless orbit $\{z_t\}_{t=1}^{50}$ with $z_1 = p_1 = 0.8724....$ Note that although z_t is close to p_t for initial times, after about 10 steps they start to diverge. On the other hand, starting with $s_1 = 0.8723...$, we found a true orbit $\{s_t\}_{t=1}^{50}$, shown in (red) crosses, which is able to match the entire observed orbit $\{p_t\}_{t=1}^{50}$. In the lower panel we show the differences between $\{p_t\}, \{z_t\}$ (in blue triangles) and $\{p_t\}, \{s_t\}$ (in red crosses) respectively. Although generated by the same model, $\{z_t\}$ and $\{s_t\}$ apparently leads to different conclusions about the model quality. In general, since there are infinitely many orbits the model can produce, how to judge the quality of the model becomes a challenging problem.

techniques will be used, to judge how good our reduced model is, in terms of its capability of producing an orbit that matches the low dimensional time series.

Specifically, there are two types of error measures introduced in this approach. First is the dimensionality reduction error η, usually measured in Euclidean norm, which accounts for the loss of information in simplifying the observation; the second error, crucial for assessing the model quality of chaotic systems, is the shadowing error ϵ, usually measured in sup norm, which corresponds to the capability of the given model to generate one orbit that matches the observed (low dimensional) time series.

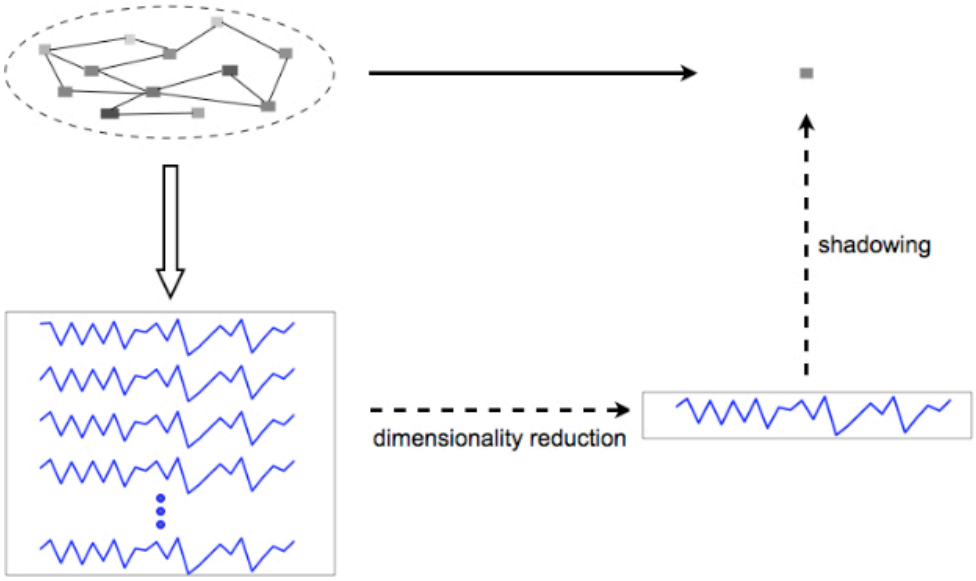

Figure 5.3. **Illustration of the model reduction process via shadowing criteria.** -

5.3 Model Reduction of Coupled Chaotic Oscillators

5.3.1 Problem Statement

To illustrate this approach, we consider the problem of judging the quality of reduced order model of a system of coupled chaotic oscillators. Since the approach involves the use of numerical shadowing, which is mostly convenient to describe (and to implement) through discrete systems, we shall consider examples where individual oscillators are governed by some discrete chaotic process.

For discrete dynamics, the coupled non-identical oscillator network can be described by:

$$w_{t+1}^{(i)} = f[w_t^{(i)}, \mu^{(i)}] - \sigma \sum_{j=1}^{n} l_{ij} f[w_t^{(j)}, \mu^{(j)}], \tag{5.2}$$

where $\{w^{(i)}\}_{i=1,\ldots,n}$ represent a set of coupled oscillators, $w_t^{(i)} \in \Re^d$ is the state of oscillator i at time t; $f : \Re^d \times \Re^p \to \Re^d$ is assumed to be a C^1 diffeomorphism describing the individual dynamics; the second term describes the effective coupling between different oscillators through a discrete Laplacian matrix $L = [l_{ij}]_{n\times n}$, with the property that for each i, $\sum_{j=1}^{n} l_{ij} = 0$; and σ is the coupling strength. The theory and methods developed in this letter can be extended to continuous systems. The coupling function f has been chosen to have the same form of the individual dynamics, which corresponds to the situation where each oscillator receives a direct signal from the output of its neighbors. The second argument $\mu^{(i)}$ in the function f plays an important role. It allows possible mismatch of parameters between different individual oscillators, which is usually the case for a physical setting.

For this high $(n \times d)$ dimensional coupled chaotic system, several questions are of particular interest, as initial exploration for the general problem as we analyzed in Chapter 2 for non chaotic systems, and will be answered in this letter. Fig. 5.4 serves as an illustration.

1 In what sense can we model a coupled identical oscillator network by a single oscillator?

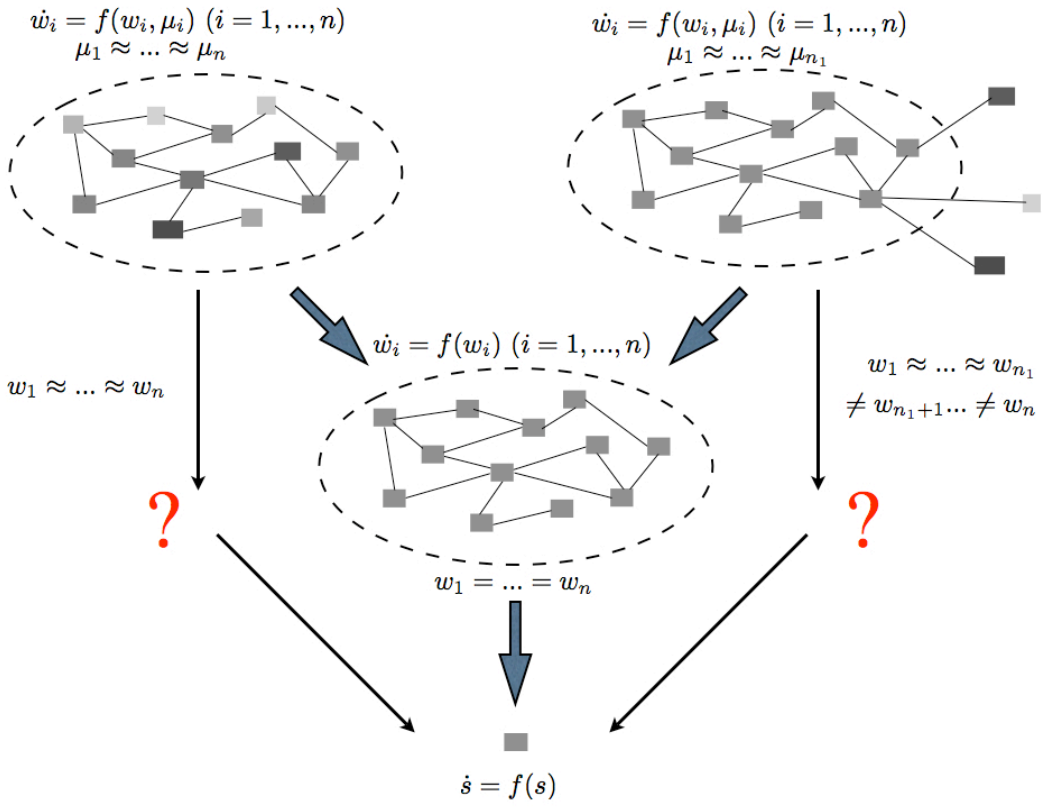

Figure 5.4. **Illustration of open questions regarding the model reduction of coupled chaotic oscillators** (5.2)**.** - In the first case (represented in the top ellipse), all the oscillators are the same; in the second case (middle ellipse) the oscillators are mismatched; while in the third case, the network consists of a cluster of identical oscillators with a few outliers. The rectangles represent individual oscillators (the width and color are set to be slightly different, to represent the difference of individual dynamics), edges connecting them indicate the presence of coupling. In case one, the system can be modeled by a single oscillator if these oscillators are *completely synchronized*. In case two and three complete synchronization is not in general possible. However, if the oscillators (in the cluster, for case three) are *nearly synchronized* (SBN09d; SBN09c), model reduction is possible, under certain conditions (will be discussed in the main context).

2 In what sense can we model a coupled *non-identical* oscillator network by a single oscillator?

3 In what sense can we model a *nearly synchronized cluster* by a single oscillator?

For question 1, a general criteria is to determine whether the system synchronizes or not. When the oscillators synchronize completely, we have:

$$\lim_{t\to\infty} ||w_t^{(i)} - w_t^{(j)}|| \to 0, \forall i, j. \tag{5.3}$$

After transient time, since $\mu_1 = ... = \mu_n \equiv \mu$, all the oscillators evolve in the same way, and the second term in Eq. (5.2) disappears (so there will be no error in the step of dimensionality reduction and shadowing). The motion of oscillator i is governed by

$$w_{t+1}^{(i)} = f[w_t^{(i)}, \mu]. \tag{5.4}$$

Eqs. (5.3) and (5.4) allows us to perfectly model the coupled system by a single, low dimensional oscillator

$$s_{t+1} = f[s_t, \mu]. \tag{5.5}$$

Questions 2 and 3 are intriguing. In these cases, the oscillators are unable to completely synchronize, thus a single oscillator model may not exactly represent the true collective behavior of the coupled system. In particular, if one chooses the average trajectory as a representative (as a low dimensional representation of the high dimensional time series), then this average orbit is governed by

$$\bar{w}_{t+1} = \frac{1}{n}\sum_{i=1}^{n} f[w_t^{(i)}, \mu^{(i)}] - \frac{\sigma}{n}\sum_{i,j=1}^{n} l_{ij} f[w_t^{(j)}, \mu^{(j)}], \tag{5.6}$$

which depends essentially on *every* single oscillator of the previous state, implying that the dimension of the system is as high as the original coupled system. Even in the situation where the oscillators are *nearly synchronized* (see Fig. 5.5 as an example), such that

$$\limsup_t ||w_t^{(i)} - \bar{w}_t|| \approx 0, \tag{5.7}$$

if one were to use mean-field approximation, for example, replace $w_t^{(i)}$ with $\bar{w}_t$ and $\mu^{(i)}$ with $\bar{\mu}$, resulting in a low dimensional model:

$$\bar{w}_{t+1} = f[\bar{w}_t, \bar{\mu}], \tag{5.8}$$

then at each step this inexact model generates error, so an trajectory from this low dimensional model usually diverges from that from the actual average trajectory due to the sentive dependence on initial conditions, a typical property of chaotic systems. Even a perfect model would suffer from this property (see Fig. 5.2 for a simplest case). It is the concept of shadowing which allows us to quantify the quality of this model appropriately.

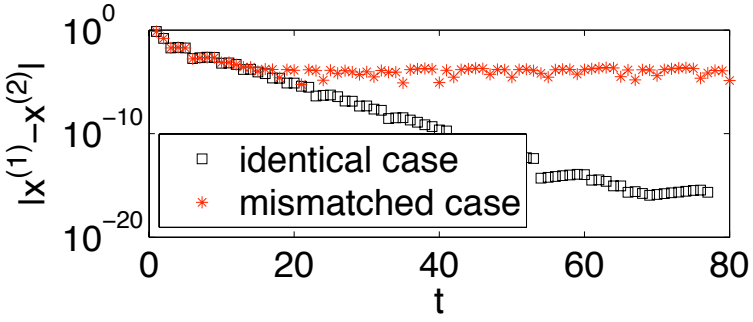

Figure 5.5. **Difference between complete (identical) and nearly synchronization.** - Consider a simple two coupled logistic maps with coupling strength $\sigma = 0.35$. In the first case both the oscillators have parameters $a = 4$, while in the second case $a_1 = 3.999$ and $a_2 = 4.001$. The (black) squares are the differences in Euclidean norm between the time series from the two orbits in case 1 (identical case), and the (red) crosses are similar plot for case 2 (mismatched case). Note that in the identical, the difference converges to 0 exponentially, which corresponds to stable complete synchronization; while in the mismatched case the difference between the two oscillators stays small, but *finite*, referred to as *nearly synchronization*.

5.3.2 Example of Coupled Logistic Maps

The approach which adopts the idea of shadowing for judging the model quality is justified in the following examples.

Consider as an example a set of mismatched logistic maps, with individual dynamics:

$$f[x_t^{(i)}, a^{(i)}] = a^{(i)} x_t^{(i)} (1 - x_t^{(i)}). \tag{5.9}$$

where we $x_t^{(i)}$ is the state variable of node i at time t, and $a^{(i)} \approx 4$ denotes its individual parameter. In practice, we restrict the state variable to belong to the interval $[0, 1]$ so that no orbit will be unbounded. This may be achieved simply by assigning value 1 to $x_t^{(i)}$ whenever its value exceeds 1.

We illustrate how the quality of model may be assessed by the approach illustrated in Section 5.2 which involves the use of shadowing.

There are two scenarios to be considered:

(1) First we illustrate how the quality of a reduced order model for a network of coupled mismatched logistic oscillators may be assessed by the idea of shadowing. We assume that the collection of the above logistic oscillators are coupled through an Erdos-Renyi network of $n = 1000$ vertices and 49893 edges with coupling strength $\sigma = 0.01$.

(2) Second we show the reduced model assessment for a synchronized cluster in a coupled identical oscillator network. In this example all the oscillators are identical, with parameters $a = 4$. The underlying network has a densely connected cluster, which is a Erdos-Renyi network of $n = 200$ vertices and $m = 3919$ edges. An outlier is constructed, by adding a new node to this network, connecting to one node in the cluster with an single edge. The coupling strength of the whole network is set as $\sigma = 0.025$, in which case the cluster is nearly synchronized, while the whole network is not.

In Fig. 5.6 we evaluate one-parameter family of reduced models $f(x) = ax(1-x)$ where a is the parameter for those two high dimensional systems by comparing their shadowing distances. Here we use a finite trajectory of length $T = 500$ after transient time. The shadowing distances are computed by the numerical technique developed in the previous chapter.

The above two examples illustrate how one can judge the quality of a given model, or compare different models quantitatively by using shadowing techniques.

In both examples we have used the average trajectory as a low dimensional representative for the high dimensional time series, since the choice of average minimizes the square distance to all other individual trajectories. However, since our goal is to model a chaotic system, sometimes this dimensionality reduction error is not as important as the shadowing error, what will change if relax the condition of choosing an optimal low dimensional representative?

Consider an example of coupled oscillator network with identical logistic maps where the parameter $a = 4$. The underlying network is designed to have a three-layer

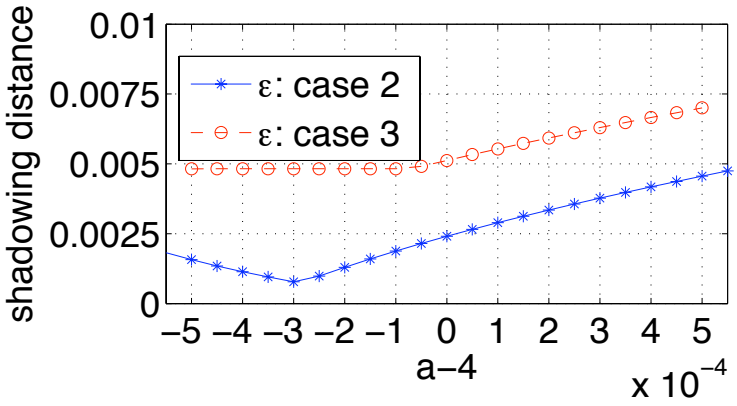

Figure 5.6. **Shadowing error of reduced order model for coupled oscillator networks.** - Blue stars correspond to the result for shadowing errors for one-parameter family of reduced models $f(x) = ax(1-x)$ for coupled mismatched logistic oscillators through a network of $n = 1000$ nodes and $m = 49893$ randomly placed edges. The coupling strength is $\sigma = 0.01$. Similarly, red circles correspond to similar reduced models for a synchronized cluster from a coupled logistic oscillator network consisting of identical logistic maps ($a = 4$). The network is made up with a cluster which has 200 nodes and 3919 random edges, and one outlier that connect to one node from the cluster.

structure, and is constructed as follows: the inner layer is a complete subgraph of $n_1 = 100$ nodes; then a middle layer is added, which is a random subgraph of $n_2 = 50$ nodes generated by the Erdos-Renyi model where any two nodes are connected with probability 0.5, also, a node from the inner layer has a link to a node from the middle layer with probability 0.5; the outer layer consists of a single node, which connects to 5% of the nodes from the middle layer. The coupling strength is set to be $\sigma = 0.01$ in this case.

To compare the dimensionality reduction error and the shadowing error for $\lambda \in [0, 1]$, a one parameter family of time series $y(\lambda) = \lambda x_1 + (1 - \lambda)x_2$ is used, where x_1 is the average time series from the inner layer, and x_2 is the average from the middle layer. The results are shown in Fig. 5.7 and Fig. 5.8. In Fig. 5.7, the monotone decrease of the curves with increasing value of λ corresponds to the fact that the inner layer is more connected and thus strongly synchronized, which leads to smaller shadowing error for larger λ. Fig. 5.8 shows that, when μ increases (i.e., more weight is given to the dimensionality reduction error than the shadowing error), the λ which corresponds to the minimum of the model reduction error decreases. In general, it is

the nature of the problem which advice the choice of μ, and result in different model criteria.

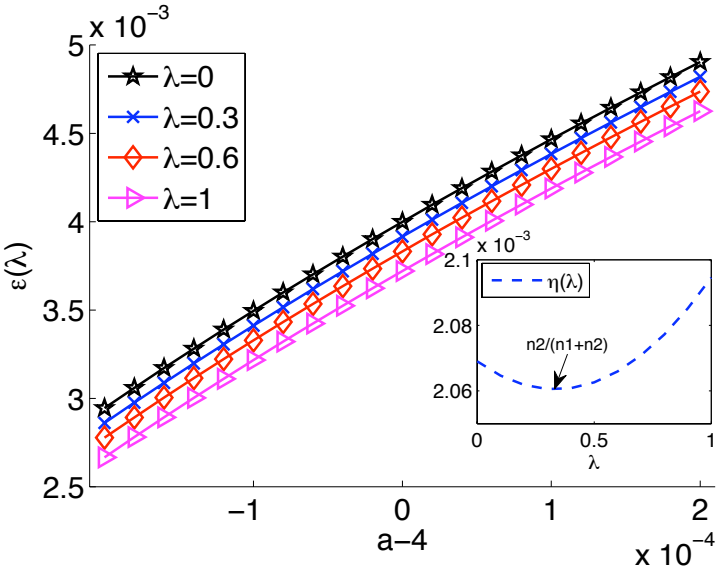

Figure 5.7. **Dependence of shadowing error on λ.** - This figure shows, for different choices of λ, the shadowing error for different models which depend on the parameter a used in the model. The inset shows the dimensionality reduction error curve with respect to λ, which is minimized at $\lambda = \frac{1}{3}$ in this case.

5.4 Discussion and Open Problems

In this chapter we propose a general approach for assessing the quality of a reduced order model for high dimensional chaotic systems. The key in this approach is the novel application of shadowing, combined with dimensionality reduction techniques. This approach overcomes the usual difficulty encountered by traditional methods which either tries to compare systems of the same size by measuring the distance in the functional space, or measure how close an orbit generated by a model is to the observed data.

We have illustrated the validity of our approach by examples of coupled Henon oscillator network in the cases where the oscillators are either nearly synchronized, or only a portion of the oscillators (not all) are synchronized. Interestingly, a parameter

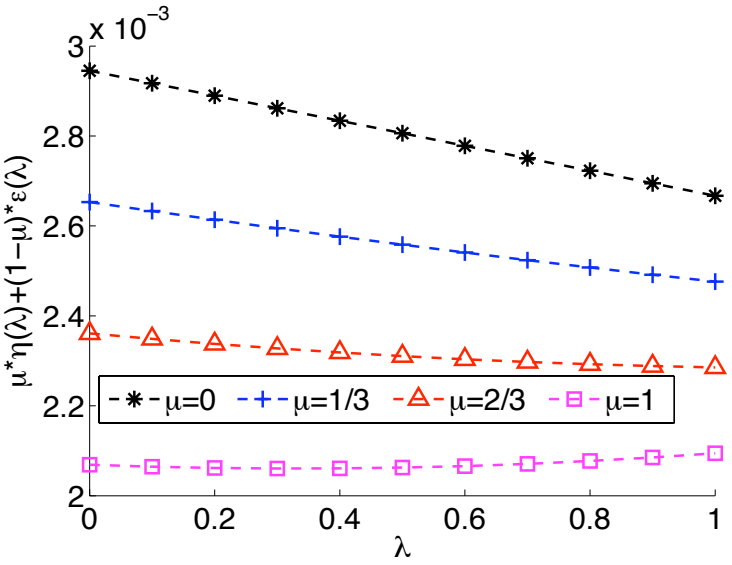

Figure 5.8. **Interplay between dimensionality reduction error and shadowing error.** - In $\mu = 0$ case, the curve is the top one, which shows that the model reduction error decreases monotonically with respect to the increase of λ; this trend remains for other choices of μ such as $\mu = \frac{1}{3}$ (the second top curve) and $\mu = \frac{2}{3}$ (the second bottom curve). However, when μ is slowly increasing to the value 1, in which case much more emphasis is given on the dimensionality reduction error than the shadowing error, the minimum of this curve (on the bottom) is no longer attained at $\lambda = 1$, but near $\lambda = \frac{1}{3}$.

shift is found for an optimal model as opposed to the direct average of individual dynamics.

Problems left open include, given this measure of how good a model is, how to efficiently find an optimal model? How to model network dynamics in some hierarchical sense. Results on such problems will be reported in future work.

Chapter 6

Dynamics of Networks: Updating Schema for Local Statistics

6.1 Introduction

Complex networks are useful tools for modeling complicated real life objects and their interactions. Examples include computer networks, social networks, biological networks, etc. (AB02; BA99; DGM08; New03; PVV01; VPV02; WS98; Zho06). Recently developed statistical methods (AB02) allow us to analyze large networks by several important macroscopic statistic meant to summarize the massive amount of information required to completely describe the network, an impossible task without the availability of modern computers. These statistics include such things as degree (number of connections each node has), clustering coefficient (WS98), assortativity coefficient (New02), modularity measure (New06), etc. Fast algorithms (Net; Paj) have been developed to compute these statistics for any given network, either represented by adjacency matrix or edge list (CLRS01).

Despite that many real world networks evolve, and even grow in time, it is only recently that network models have allowed for this ubiquitous feature. Understanding the evolution of such networks invariably leads quickly to computing network statistics as they evolve in time, (KW06; GBB09). In this regard, for any evolving network, to

measure the corresponding evolution of network statistics, the computation based on static network structure must be done (for the network) at each time step, resulting in a costly (and perhaps, impractical) computation. A missing part in the study of evolving networks is a development of a dynamic algorithm which updates the statistics rather than recomputing the quantity from scratch.

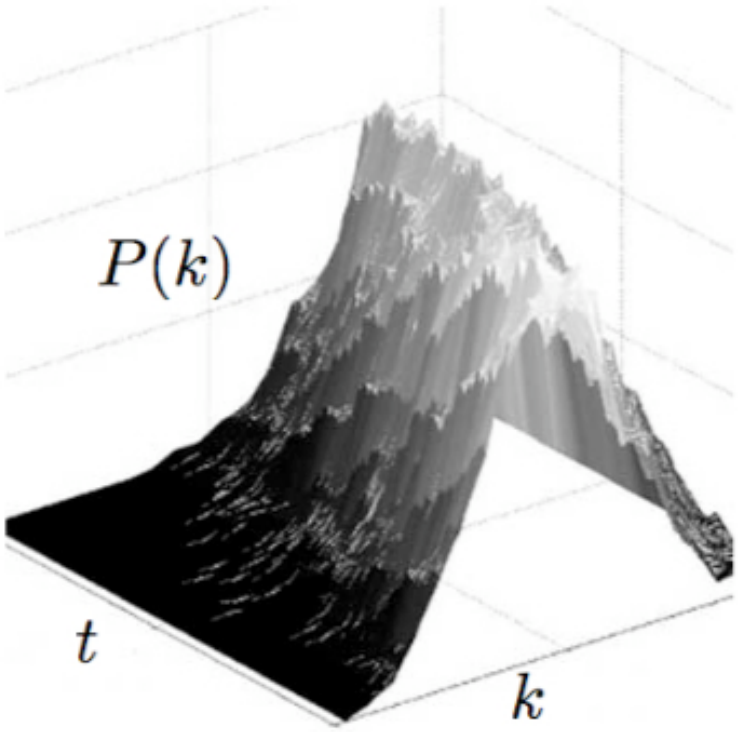

Figure 6.1. **Evolution of the degree distribution of a randomly growing network.** - This picture shows the computed evolution of degree distribution of an evolving network constructed by simply adding edges at random, one at a time. This computation was performed using the algorithm developed in this chapter, and would be impractical if one were to do direct computation. The figure is adopted from (SBBS09), and can also be found at the Physical Review E Kaleidoscope Images (March 2009): http://pre.aps.org/kaleidoscope/March2009.

In this chapter we present an update algorithm specifically for local statistics based on the knowledge of existing network structure and the changes to the network. Our methods allow us to update relevant statistics of a large network by considering only the adjustments to the network. (See Fig. 6.1 for an example of the application of our algorithm). This philosophy represents a significant theoretical advancement in understanding large scale networks (critical quantities and topological features), and

it also leads to efficient algorithms for tracking evolution of large scaled networks in time.

Although it is relatively straightforward to formulate the update schema for local statistics once one realizes that updating is usually better than re-computing, it becomes challenging to develop similar update schema for global statistics such as shortest paths and spectrum. Such problems will be left for discussion in the next chapter.

The rest of Chapter 6 is organized as follows: In Section 6.2, we review the definition of some network statistics and introduce notation that will be used in the chapter for these statistics. In Section 6.3, we derive update formulas for network statistics upon the change of network structure and we will compare the computational complexity to the use of standard methods. In Section 6.4, we provide a simplified, algorithmic representation of our update scheme; and in Section 6.5, we show examples of application of the update scheme. In Section 6.6, we discuss the main results of this chapter and give some overview of potential future research.

6.2 Local Graph Statistics

We represent a network using a graph $G = (V, E)$ where $V = \{1, 2, ..., n\}$ is the vertex set and $E = \{(i, j) \,|\, i \text{ and } j \text{ are connected}\}$ is the edge set. Note that for undirected graphs, it is not necessary to include (j, i) in the set E if $(i, j) \in E$, however, including both of them allows easier generalization to directed graphs. In this work, we limit ourselves to undirected, unweighted networks; their graphs possess a symmetric, binary *adjacency matrix* A:

$$a_{ij} = \begin{cases} 1, & \text{if } (i,j) \in E; \\ 0, & \text{otherwise.} \end{cases} \tag{6.1}$$

Let m denote the total number of edges in G. Then

$$m \equiv \frac{1}{2}|E| = \frac{1}{2}||A||_F^2 = \frac{1}{2}\sum_{i,j} a_{ij}, \tag{6.2}$$

where $|.|$ is the cardinality of a set.

Define the *neighborhood* $N(i)$ of node i as the set of vertices that are adjacent to i:

$$N(i) \equiv \{j|(i,j) \in E\} = \{j|a_{ij} = 1\}. \tag{6.3}$$

Likewise, define the *shared* neighborhood N_{ij} of nodes i and j as:

$$N_{ij} \equiv N(i) \cap N(j). \tag{6.4}$$

The *degree* k_i of node i is the number of nodes i is adjacent to:

$$k_i \equiv |N(i)| = \sum_j a_{ij} = \sum_j a_{ji}. \tag{6.5}$$

The *clustering coefficient* of node i is defined by (WS98):

$$C_i \equiv \begin{cases} \frac{2\triangle_i}{k_i(k_i-1)}, & \text{if } k_i \geq 2; \\ 0, & \text{otherwise,} \end{cases} \tag{6.6}$$

where $\triangle_i$ is the number of triangles that contain i. Then the *average clustering coefficient* [1] of the whole network is simply the average of all C_i's:

$$C \equiv \frac{1}{n} \sum_i C_i. \tag{6.7}$$

The *assortativity coefficient* r, which describes the correlation of the degree of adjacent nodes of a network(New02), is computed as:

$$\begin{aligned} r &\equiv \frac{8m \sum_{(i,j)\in E} k_i k_j - \left[\sum_{(i,j)\in E} (k_i + k_j)\right]^2}{4m \sum_{(i,j)\in E} \left(k_i^2 + k_j^2\right) - \left[\sum_{(i,j)\in E} (k_i + k_j)\right]^2} \\ &= \frac{8mu - v^2}{4mw - v^2}, \end{aligned} \tag{6.8}$$

where

$$u \equiv \sum_{(i,j)\in E} k_i k_j, \tag{6.9}$$

$$v \equiv \sum_{(i,j)\in E} (k_i + k_j), \tag{6.10}$$

$$w \equiv \sum_{(i,j)\in E} \left(k_i^2 + k_j^2\right). \tag{6.11}$$

[1] There is an alternative definition of the average clustering coefficient of a network in (New03), and the formula for updating this alternative C can be derived easily based on updating the number of triangles and triples in the network.

Modularity Q (New06) measures the quality of a community partition and is typically defined as:

$$\begin{aligned} Q &\equiv \frac{1}{2m}\sum_{i,j}\left(a_{ij} - \frac{k_i k_j}{2m}\right)\delta(g_i, g_j) \\ &= \frac{1}{2m}\left[S_A - \frac{1}{2m}S_P\right], \end{aligned} \tag{6.12}$$

where $\delta(g_i, g_j) = 1$ if nodes i and j are in the same group and zero otherwise, and

$$S_A \equiv \sum_{i,j} a_{ij}\delta(g_i, g_j), \quad S_P \equiv \sum_{i,j} k_i k_j \delta(g_i, g_j). \tag{6.13}$$

Note that all the above statistics (degree, clustering coefficient, assortativity coefficient, and modularity) depend on some function of the statistics of individual nodes in a *local* sense. This observation is key to the efficient update of those statistics.

6.3 Updating Local Statistics

6.3.1 Connecting a New Node

The operation of adding an edge to a new node can be decomposed into two successive operations. First, introduce an isolated node that connects to nothing in the network; then add an edge between this node and a previously existing node. In this subsection we discuss the effect of adding an isolated node, and leave the discussion of adding an edge to the next subsection. We use $\widetilde{\cdot}$ to represent updated statistics. [Note: this section provides the derivation of update formulas. For algorithmic representation of this scheme, the reader should look to Section 6.4.]

Since no new edge is introduced, it is easy to obtain the following updating relations:

$$\widetilde{n} = n + 1, \quad \widetilde{m} = m, \quad \widetilde{E} = E, \tag{6.14}$$

and

$$\widetilde{a}_{ij} = \begin{cases} a_{ij}, & \text{if } i \neq n+1 \text{ and } j \neq n+1; \\ 0, & \text{otherwise.} \end{cases} \tag{6.15}$$

Then for other statistics, we have:

$$\begin{aligned} \widetilde{k}_i &= k_i, \quad i \neq n+1; \\ \widetilde{k}_{n+1} &= 0; \end{aligned} \tag{6.16}$$

and

$$\begin{aligned} \widetilde{C}_i &= C_i, \quad i \neq n+1; \\ \widetilde{C}_{n+1} &= 0, \end{aligned} \tag{6.17}$$

so that

$$\widetilde{C} = \frac{1}{\widetilde{n}} \sum_i \widetilde{C}_i = \frac{1}{n+1} \sum_i C_i = \frac{n}{n+1} C. \tag{6.18}$$

Similarly, $\widetilde{r} = r$ since $\widetilde{u} = u$, $\widetilde{v} = v$, and $\widetilde{w} = w$; and $\widetilde{Q} = Q$ since $\widetilde{S_A} = S_A$ and $\widetilde{S_P} = S_P$.

6.3.2 Adding an Edge between Existing Nodes

Suppose $a_{pq} = 0$ ($p \neq q$ and p, q are not connected), we analyze the impact of connecting p and q on the various statistics of the network. See Fig. 6.2 for an example.

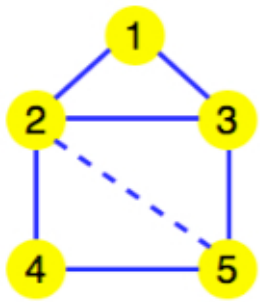

Figure 6.2. **Schematic addition of an edge.** - The original graph consists of $n = 5$ nodes (solid circles) and $m = 6$ edges (solid lines). For the original graph the degree vector is $k = [2, 3, 3, 2, 2]$, while the triangle vector is $\triangle = [1, 1, 1, 0, 0]$, the clustering coefficient vector is $C = [1, \frac{1}{3}, \frac{1}{3}, 0, 0]$, and the average clustering coefficient is $\frac{1}{3}$. After an edge is added between node 2 and node 5 (dashed line), the above statistics change to $\widetilde{k} = [2, 4, 3, 2, 3]$, $\widetilde{\triangle} = [1, 3, 2, 1, 2]$, $\widetilde{C} = [1, \frac{1}{2}, \frac{2}{3}, 1, \frac{2}{3}]$, and the new average clustering coefficient becomes $\frac{23}{30}$.

The goal is to derive computations that are as inexpensive as possible. We use $\widetilde{\cdot}$ to represent updated statistics:

$$\widetilde{E} = E \cup \{(p,q),(q,p)\}, \tag{6.19}$$

$$\widetilde{m} = m + \Delta^+ m = m + 1, \tag{6.20}$$

and

$$\widetilde{a}_{ij} = a_{ij} + \Delta^+ a_{ij} = a_{ij} + \delta_{ip}\delta_{jq} + \delta_{iq}\delta_{jp}, \tag{6.21}$$

where we use *update delta* Δ^+ to represent the change for statistics upon adding an edge to the existing network. In the following part the notation Δ^- will also be used, to denote change for statistics upon deleting an existing edge. When there is no confusion, we will not explicitly specify which edge to add or delete in the update delta notation.

Based on the above formulas, we can derive schemes for efficiently updating network statistics.

Degree(+)

The change in degree for node i is simply:

$$\widetilde{k}_i = k_i + \Delta^+ k_i = k_i + \delta_{ip} + \delta_{iq}, \tag{6.22}$$

where

$$\Delta^+ k_i = \delta_{ip} + \delta_{iq}. \tag{6.23}$$

The above formula indicates that the degree changes only for vertex p and q. Thus if we keep a list of the degree of all vertices of the network, each update takes only two operations when a new edge is added.

Clustering Coefficient(+)

To compute the new clustering coefficient of each node, and thus the whole network, we need the updated number of triangles at node i:

$$\widetilde{\triangle}_i = \begin{cases} \triangle_i, & \text{if } i \notin \{p,q\} \cup N_{pq}; \\ \triangle_i + 1, & \text{if } i \in N_{pq}; \\ \triangle_i + |N_{pq}|, & \text{if } i \in \{p,q\}. \end{cases} \tag{6.24}$$

Combining this with Eq. (6.22) and $\triangle_i = \frac{1}{2}C_i k_i(k_i-1)$, from Eq. (6.6), we have:

$$\widetilde{C}_i = \begin{cases} C_i, & \text{if } i \notin \{p,q\} \cup N_{pq}; \\ C_i + \frac{2}{k_i(k_i-1)}, & \text{if } i \in N_{pq}; \\ \frac{k_i-1}{k_i+1}C_i + \frac{2|N_{pq}|}{k_i(k_i+1)}, & \text{if } i \in \{p,q\}. \end{cases} \tag{6.25}$$

Note that whenever the denominator of a fraction is zero, we define the fraction to be zero, such as in Eq. (6.25) and also throughout. This maintains the consistency that $C_i = 0$ if $k_i < 2$. Finally, the average clustering coefficient C becomes: $\widetilde{C} = C + \Delta^+ C$ where

$$\Delta^+ C = \frac{2}{n}\left[\sum_{i \in N_{pq}} \frac{1}{k_i(k_i-1)} + \sum_{i \in \{p,q\}} \left(\frac{|N_{pq}|}{k_i(k_i+1)} - \frac{C_i}{k_i+1}\right)\right]. \tag{6.26}$$

Note that to update the average clustering coefficient, we need to keep the clustering coefficient for each node in order to apply the update formula. This implies an $O(n)$ storage complexity.

Assortativity Coefficient(+)

To compute $\widetilde{r}$, we need $\widetilde{u}$, $\widetilde{v}$, and $\widetilde{w}$. The update formula for u is:

$$\begin{aligned} \widetilde{u} &= \sum_{(i,j)\in\widetilde{E}} \widetilde{k}_i \widetilde{k}_j = \sum_{(i,j)\in E} \widetilde{k}_i \widetilde{k}_j + 2(k_p+1)(k_q+1) \\ &= \sum_{(i,j)\in\widehat{E}} k_i k_j + 2\sum_{i\in N(p)} k_i(k_p+1) \\ &\quad + 2\sum_{i\in N(q)} k_i(k_q+1) + 2(k_p+1)(k_q+1) \\ &= u + 2\left(\sum_{i\in N(p)} k_i + \sum_{i\in N(q)} k_i\right) + 2(k_p+1)(k_q+1) \\ &= u + \Delta^+ u. \end{aligned} \tag{6.27}$$

Here $\widehat{E} = E \setminus \{(p,q),(q,p)\}$ is the edge set that contains all edges in E except(p,q) and (q,p), and

$$\Delta^+ u = 2\left(\sum_{i\in N(p)} k_i + \sum_{i\in N(q)} k_i\right) + 2(k_p+1)(k_q+1). \tag{6.28}$$

Similarly, we can obtain update formula for v and w:

$$\begin{aligned}\widetilde{v} &= \sum_{(i,j)\in\widetilde{E}} (\widetilde{k}_i + \widetilde{k}_j) \\ &= v + 4(k_p + k_q + 1) = v + \Delta^+ v,\end{aligned} \tag{6.29}$$

where

$$\Delta^+ v = 4(k_p + k_q + 1). \tag{6.30}$$

For w we have:

$$\begin{aligned}\widetilde{w} &= \sum_{(i,j)\in\widetilde{E}} (\widetilde{k}_i^2 + \widetilde{k}_j^2) \\ &= w + \Delta^+ w,\end{aligned} \tag{6.31}$$

where

$$\Delta^+ w = 6\left[k_p(k_p+1) + k_q(k_q+1)\right] + 4. \tag{6.32}$$

Finally, the new assortativity coefficient can be updated using:

$$\begin{aligned}\widetilde{r} &= r + \Delta^+ r \\ &= \frac{8\widetilde{m}\widetilde{u} - \widetilde{v}^2}{4\widetilde{m}\widetilde{w} - \widetilde{v}^2} \\ &= \frac{8\left(m+1\right)\left(u + \Delta^+ u\right) - \left(v + \Delta^+ v\right)^2}{4\left(m+1\right)\left(w + \Delta^+ w\right) - \left(v + \Delta^+ v\right)^2}.\end{aligned} \tag{6.33}$$

Modularity(+)

For modularity, we assume that after connecting the nodes p and q, the partitions g_i do not change for any node i. Then the new modularity measure will be:

$$\widetilde{Q} = \frac{1}{2\widetilde{m}}\left[\widetilde{S}_A - \frac{1}{2\widetilde{m}}\widetilde{S}_P\right]. \tag{6.34}$$

We already have $\widetilde{m} = m + 1$, so we can now derive updating formulas for S_A and S_P. By Eq. (6.13), we have:

$$\begin{aligned}\widetilde{S}_A &= S_A + \Delta^+ S_A \\ &= \sum_{i,j} \widetilde{a}_{ij}\delta(g_i, g_j) \\ &= \sum_{i,j} \left(a_{ij} + \delta_{ip}\delta_{jq} + \delta_{iq}\delta_{jp}\right)\delta(g_i, g_j) \\ &= S_A + 2\delta(g_p, g_q)\end{aligned} \tag{6.35}$$

where $\Delta^+ S_A = 2\delta(g_p, g_q)$; and

$$\begin{aligned}
\widetilde{S_P} &= S_P + \Delta^+ S_P \\
&= \sum_{i,j} \widetilde{k}_i \widetilde{k}_j \delta(g_i, g_j) \\
&= \sum_{i,j} \left(k_i + \delta_{ip} + \delta_{iq}\right)\left(k_j + \delta_{jp} + \delta_{jq}\right)\delta(g_i, g_j) \\
&= S_P + 2\sum_i k_i \big[\delta(g_i, g_p) + \delta(g_i, g_q)\big] + 2\big[\delta(g_p, g_q) + 1\big].
\end{aligned} \tag{6.36}$$

However, computing the sum in Eq. (6.36) for every update is expensive. To avoid this, define the following auxiliary statistics:

$$K_g \equiv \sum_i k_i \delta(g_i, g) \tag{6.37}$$

with the updating scheme

$$\begin{aligned}
\widetilde{K}_g &= K_g + \Delta^+ K_g \\
&= K_g + \delta(g_p, g) + \delta(g_q, g)
\end{aligned} \tag{6.38}$$

giving

$$\widetilde{S}_P = S_P + \Delta^+ S_P = S_P + 2\left(K_{g_p} + K_{g_q}\right) + 2\big[\delta(g_p, g_q) + 1\big] \tag{6.39}$$

where $\Delta^+ S_P = 2\left(K_{g_p} + K_{g_q}\right) + 2\big[\delta(g_p, g_q) + 1\big]$.

Finally, combining (6.35) and (6.39) with (6.34) gives the updating scheme for Q:

$$\begin{aligned}
\widetilde{Q} &= Q + \Delta^+ Q \\
&= \frac{1}{2(m+1)}\Big[S_A + 2\delta(g_p, g_q) - \frac{1}{2(m+1)}\Big(S_p + 2[K_{g_p} + K_{g_q}] + 2[\delta(g_p, g_q) + 1]\Big)\Big].
\end{aligned} \tag{6.40}$$

From Eq. (6.40) one is able to predict whether the modularity measure Q increases or decreases with the knowledge of the existing partition of the graph, as well as the edge to be added. For example, if there is a preexisting partition of the graph into two groups, and if a new edge is added in between the two groups, then $\Delta^+ Q < 0$, i.e., the modularity decreases. On the other hand, if a new edge is added to vertices belonging to the same group, then the modularity increases if the edge is added to the group with smaller total degree. However, adding an edge within a group does not necessarily increase Q if the edge is added into a group with a larger total degree, see Fig. 6.3 as an example.

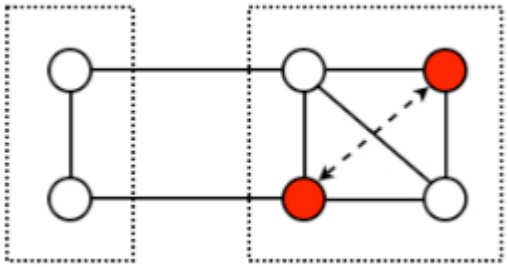

Figure 6.3. **Modularity change upon the addition of an edge.** - An example that the modularity actually decreases when a new edge is added to vertices within the same group. The dashed oval boxes indicate the preexisting partition of the graph into two groups. Solid lines are the edges in the original graph. Before adding the new edge (dashed arrow), the modularity is 0.125. After a new edge is added between two vertices in the same group (solid circles) the updated modularity becomes 0.1235.

6.3.3 Deleting an Existing Edge

Now we investigate how network statistics change when we delete an existing edge in the network. See Fig. 6.4 for an example.

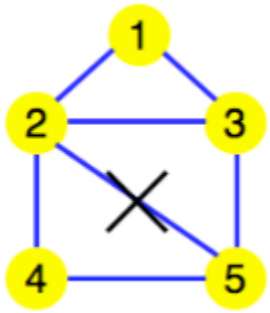

Figure 6.4. **Schematic removal of an edge.** - The original graph consists of $n = 5$ nodes (solid circles) and $m = 7$ edges (solid lines), with statistics $k = [2, 4, 3, 2, 3]$, $\triangle = [1, 3, 2, 1, 2]$, $C = [1, \frac{1}{2}, \frac{2}{3}, 1, \frac{2}{3}]$, and the average clustering coefficient $\frac{23}{30}$. After the edge between node 2 and node 5 is deleted, the new statistics are: $\widehat{k} = [2, 3, 3, 2, 2]$, $\widehat{\triangle} = [1, 1, 1, 0, 0]$ $\widehat{C} = [1, \frac{1}{3}, \frac{1}{3}, 0, 0]$, and the new average clustering coefficient is $\frac{1}{3}$. Note that this is a symmetric operation to the one shown in Fig. 6.2.

Suppose $a_{pq} = 1$ ($p \neq q$ and p, q are connected), and we delete the edge $(p, q) \cup (q, p)$, from our edge set E. We will use $\widehat{A}$ to represent the updated adjacency matrix,

and similarly for other statistics. Then we immediately have:

$$\widehat{E} = E \setminus \{(p,q),(q,p)\}, \tag{6.41}$$

$$\widehat{m} = m - 1, \tag{6.42}$$

and

$$\widehat{a}_{ij} = a_{ij} + \Delta^- a_{ij} = a_{ij} - \delta_{ip}\delta_{jq} - \delta_{iq}\delta_{jp.} \tag{6.43}$$

Degree(−)

The change in degree for node i is: $\widehat{k}_i = k_i + \Delta^- k_i$ where

$$\Delta^- k_i = -\delta_{ip} - \delta_{iq}. \tag{6.44}$$

Clustering Coefficient(−)

For the new clustering coefficient, we first obtain the formula for updating the number of triangles containing node i:

$$\widehat{\triangle}_i = \begin{cases} \triangle_i, & \text{if } i \notin \{p,q\} \cup N_{pq}; \\ \triangle_i - 1, & \text{if } i \in N_{pq}; \\ \triangle_i - |N_{pq}|, & \text{if } i \in \{p,q\}. \end{cases} \tag{6.45}$$

Then we obtain the formula for updating C_i:

$$\widehat{C}_i = \begin{cases} C_i, & \text{if } i \notin \{p,q\} \cup N_{pq}; \\ C_i - \frac{2}{k_i(k_i-1)}, & \text{if } i \in N_{pq}; \\ \frac{k_i}{k_i-2}C_i - \frac{2|N_{pq}|}{(k_i-1)(k_i-2)}, & \text{if } i \in \{p,q\}. \end{cases} \tag{6.46}$$

The average clustering coefficient C is updated by: $\widehat{C} = C + \Delta^- C$ where where

$$\Delta^- C = -\frac{2}{n}\left[\sum_{i \in N_{pq}} \frac{1}{k_i(k_i-1)} + \sum_{i \in \{p,q\}} \left(\frac{|N_{pq}|}{(k_i-1)(k_i-2)} - \frac{C_i}{k_i-2}\right)\right]. \tag{6.47}$$

Assortativity Coefficient(−)

The updating formulas for u, v, w are $\widehat{u} = u + \Delta^- u, \widehat{v} = v + \Delta^- v, \widehat{w} = w + \Delta^- w$, where

$$\begin{aligned} \Delta^- u &= -2\left(\sum_{i \in N(p)} k_i + \sum_{i \in N(q)} k_i\right) - 2(k_p - 1)(k_q - 1), \\ \Delta^- v &= -4\left(k_p + k_q - 1\right), \\ \Delta^- w &= -6\left[k_p(k_p-1) + k_q(k_q-1)\right] - 4. \end{aligned} \tag{6.48}$$

Then the new assortativity coefficient $\widehat{r}$ is given by

$$\widehat{r} = \frac{8\widehat{m}\widehat{u} - \widehat{v}^2}{4\widehat{m}\widehat{w} - \widehat{v}^2} = \frac{8(m-1)(u + \Delta^- u) - (v + \Delta^- v)^2}{4(m-1)(w + \Delta^- w) - (v + \Delta^- v)^2}. \tag{6.49}$$

Modularity(−)

For modularity, we again assume that the community partitions g_i are unchanged after disconnecting the edge between p and q. It follows that

$$\widehat{S}_A = S_A + \Delta^- S_A = S_A - 2\delta(g_p, g_q), \quad , \tag{6.50}$$

$$\widehat{S}_P = S_P + \Delta^- S_P = S_P - 2\left(K_{g_p} + K_{g_q}\right) + 2\left[\delta(g_p, g_q) + 1\right] \tag{6.51}$$

where K_g is now updated using

$$\widehat{K}_g = K_g + \Delta^- K_g = K_g - \delta(g_p, g) - \delta(g_q, g). \tag{6.52}$$

These now define the updating scheme for $\widehat{Q} = \left(\widehat{S}_A - \widehat{S}_P/2\widehat{m}\right)/2\widehat{m}$.

We remark here that the operation of removing a node can be decomposed into the removal of edges that node is adjacent to. Thus, formulae given for the deletion of an edge can also be used for the removal of a node.

6.4 Algorithmic Representation and Complexity

6.4.1 Algorithmic Representation of the Update Schema

In this section We give pseudo-code for implementing our update schema upon single change to the network, as described in Section 6.3.

Pseudo code of the update schema for local statistics

Given: A, a symmetric binary matrix corresponding to a simple graph.

Compute initial statistics:

n: number of nodes, m: number of edges;

k: degree vector, C: clustering coefficient vector, $\bar{C}$: average clustering coefficient;

u, v, w, r: assortativity coefficient and related statistics, as described in Section 6.2;

S_A, S_P, g, K, Q: modularity measure and related statistics, as described in Section 6.2, and in Eq. (6.37).

if a new node is added to the network without any connection **then**

$C \to [C; 0]$, $\bar{C} = \frac{n}{n+1}\bar{C}$;

$A \to [A, 0; 0]$;

$n \to n + 1$;

$k \to [k; 0]$.

---

else if a new edge is added between nodes p and q **then**

---

(1) *Update C*:

Set: $\Delta^+ C = 0$,

Let N_{pq} denote the set of common neighbors of p and q, and

Set: $C_p = C_p - \frac{2}{k_p+1}C_p + \frac{2|N_{pq}|}{k_p(k_p+1)}$, $\Delta^+ C = \Delta^+ C - \frac{2}{k_p+1}C_p + \frac{2|N_{pq}|}{k_p(k_p+1)}$,

Set: $C_q = C_q - \frac{2}{k_q+1}C_q + \frac{2|N_{pq}|}{k_q(k_q+1)}$, $\Delta^+ C = \Delta^+ C - \frac{2}{k_q+1}C_q + \frac{2|N_{pq}|}{k_q(k_q+1)}$,

for every $i \in N_{pq}$ **do**

Set: $C_i = C_i + \frac{2}{k_i(k_i-1)}$ $\Delta^+ C = \Delta^+ C + \frac{2}{k_i(k_i-1)}$,

end for

Update the average clustering coefficient: $\bar{C} = \bar{C} + \frac{\Delta^+ C}{n}$.

---

(2) *Update r*:

Set: $u = u + 2(\sum_{i \in N(p)} k_i + \sum_{i \in N(q)} k_i) + 2(k_p + 1)(k_q + 1)$,

Set: $v = v + 4(k_p + k_q + 1)$,

Set: $w = w + 6[k_p(k_p + 1) + k_q(k_q + 1)] + 4$,

Update the assortativity coefficient: $r = \frac{8(m+1)u - v^2}{4(m+1)w - v^2}$.

---

(3) *Update Q*:

Set: $S_A = S_A + 2\delta(g_p, g_q)$,

Set: $S_P = S_P + 2(K_{g_p} + K_{g_q}) + 2[\delta(g_p, g_q) + 1]$,

Set: $K_g = K_g + \delta(g_p, g) + \delta(g_q, g)$,

Update the modularity measure: $Q = \frac{1}{2(m+1)}[S_A - \frac{1}{2(m+1)}S_P]$.

---

(4) *Update elementary statistics*:

Set: $A(p, q) = A(q, p) = 1$,

$m \to m + 1$,

Set: $k_p = k_p + 1, k_q = k_q + 1$.

---

else if an existing edge between nodes p and q is removed **then**

---

(1) *Update C*:

Set: $\Delta^- C = 0$,

Let N_{pq} denote the set of common neighbors of p and q, and

Set: $C_p = C_p + \frac{2}{k_p - 2}C_p - \frac{2|N_{pq}|}{(k_p-1)(k_p-2)}$, $\Delta^- C = \Delta^- C + \frac{2}{k_p-2}C_p - \frac{2|N_{pq}|}{(k_p-1)(k_p-2)}$,

Set: $C_q = C_q + \frac{2}{k_q-2}C_q - \frac{2|N_{pq}|}{(k_q-1)(k_q-2)}$, $\Delta^- C = \Delta^- C + \frac{2}{k_q-2}C_q - \frac{2|N_{pq}|}{(k_q-1)(k_q-2)}$,

for every $i \in N_{pq}$ **do**

Set: $C_i = C_i - \frac{2}{k_i(k_i-1)}$ $\Delta^- C = \Delta^- C - \frac{2}{k_i(k_i-1)}$,

end for

Update the average clustering coefficient: $\bar{C} = \bar{C} + \frac{\Delta^- C}{n}$.

---

(2) *Update r*:

Set: $u = u - 2(\sum_{i \in N(p)} k_i + \sum_{i \in N(q)} k_i) - 2(k_p - 1)(k_q - 1)$,

Set: $v = v - 4(k_p + k_q - 1)$,

Set: $w = w - 6[k_p(k_p - 1) + k_q(k_q - 1)] - 4$,

Update the assortativity coefficient: $r = \frac{8(m-1)u - v^2}{4(m-1)w - v^2}$.

---

(3) *Update Q*:

Set: $S_A = S_A - 2\delta(g_p, g_q)$,

Set: $S_P = S_P - 2(K_{g_p} + K_{g_q}) + 2[\delta(g_p, g_q) + 1]$,

Set: $K_g = K_g - \delta(g_p, g) - \delta(g_q, g)$,

Update the modularity measure: $Q = \frac{1}{2(m-1)}[S_A - \frac{1}{2(m-1)}S_P]$.

---

(4) *Update elementary statistics:*

Set: $A(p,q) = A(q,p) = 0$;

$m \to m-1$;

Set: $k_p = k_p - 1, k_q = k_q - 1$.

---

end if

6.4.2 On Computational Complexity

In Table. 6.1 we compare the computational complexity of using the updating scheme and regular methods. For the update scheme, we assume that the initial statistics are already known, so that the update value listed indicates the computations required to perform a single update to those statistics. For the standard methods, we note that the operation count depends on the data structure used to represent the network. The updating scheme requires $O(1)$ operations to update for sparse graphs and at most $O(< k >)$, which requires significantly less work than regular methods when graph size becomes large.

Table 6.1. Comparison of Computational Complexity

Statistics	Adjacency Matrix	Edge List	Updating Scheme
degree (one node)	$O(n)$	$O(< k >)$	$O(1)$
degree (network)	$O(n^2)$	$O(< k > n)$	$O(1)$
clustering coefficient (one node)	$O(< k > n)$	$O(< k >^3)$	$O(< k >)$
clustering coefficient (network)	$O(< k > n^2)$	$O(< k >^3 n)$	$O(< k >)$
assortativity coefficient	$O(n^2)$	$O(< k > n)$	$O(< k >)$
modularity measure	$O(n^2)$	$O(< k > n)$	$O(1)$

Our primary focus is developing efficient algorithms for applications to problems of dynamic networks, and the computation savings is significant. For example, given a network of n vertices and m edges, if one edge is added to this network, the computation of network statistics needs to be remade using traditional methods (corresponding to columns 2 or 3 in Table. 6.1); On the other hand, our update scheme provides an efficient way to update these statistics which requires a far less number of operations (corresponding to column 1 in Table. 6.1).

One may also consider the process of building a network, which can be viewed simply as an edge-adding algorithm from a starting set of a graph with N nodes and no edges. It takes $\frac{<k>n}{2}$ steps to create the network. The update formulas for the degree and modularity indicate that computing the entire time sequence of statistics has the same computational complexity as doing the single computation for the final state (using the edge list). In the formula for the clustering coefficient it is more efficient to calculate each value along the way rather than doing a single computation of the final state, although we also need additional storage to track the clustering coefficient for each node. Computing the entire time vector of assortativity coefficients requires an additional factor $< k >$ computations, which is (typically) a minor price.

6.5 Examples of Application

In this section we show implementation of the above formula to obtain the evolution of some network statistics. We will focus on the case of adding edges between existing nodes, the other two operations will be very similar. The statistics we will calculate are the degree distribution, the average clustering coefficient and the modularity measure, although again, the evolution of other statistics can be obtained in the same manner by using the updating scheme. The evolving network models we choose are not intended to mimic real-world nets, but to show the efficiency of the updating scheme.

6.5.1 Evolution of Degree and Clustering Coefficient

In Fig. 6.5 we show the evolution of the degree distribution of a typical realization of a growing random graph (Bol01), obtained as follows: start with a random graph of $n = 1000$ nodes, with average degree $< k >= 10$. At each time step, randomly choose two nodes that are not connected, and make an edge between them, until the average degree of the network reaches $\widetilde{< k >} = 20$. The total number of time steps is 5000, which is $O(n)$ in this case. Note that using the updating scheme to obtain the evolution of degree in this case requires $O(n^2)$ operations (mostly for

initial calculation) while using regular method would require $O(n^3)$ operations (using adjacency matrix).

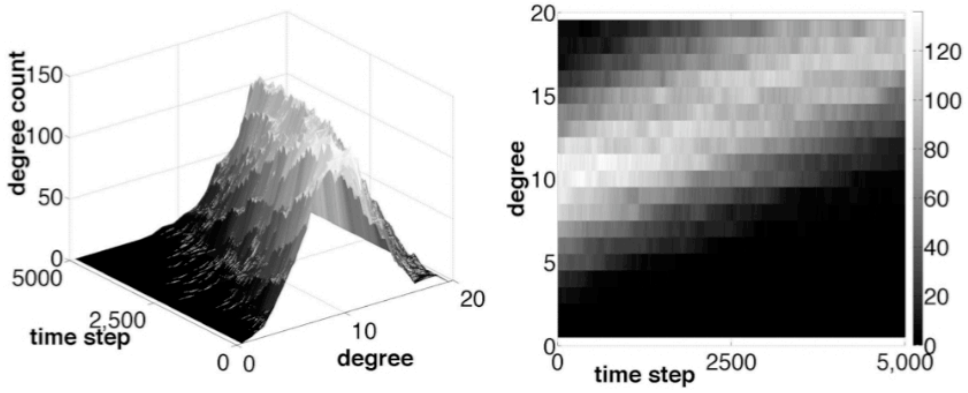

Figure 6.5. **Evolution of degree distribution of a random growing network.** - The number of vertices is 1000 in the network. Initially the connection probability of any pair of edge is 0.01, by adding random edges in the network, this probability increases to 0.02 in the end. We show two views of the evolution of the degree distribution as with respect to the process of add successive random edges. In the left panel we see that for any given time, the empirical distribution shows the underlying Poission process, and the peak is moving to larger degree side as time increases. The right panel simply provides an alternate view of the same data, providing clearer visualization of hidden portions of the 3-d surface.

In Fig. 6.6 we show the evolution of the average clustering coefficient of a typical realization of a Barabasi-Albert network (BA99). The initial network is a random network with $n = 100$ vertices and every pair of vertices is connected with probability $p = 0.5$. Then, 5000 new vertices are added in the current network successively. Each time a new vertex is introduced, it connects to two preexisting vertices according to the preferential attachment rule (BA99), which corresponds to two time steps shown in Fig. 6.6. The update scheme allows us to efficiently compute the evolution curve shown in Fig. 6.6, instead of recomputing the average clustering coefficient at each time step, which would have raised the computational requirement by about five orders of magnitude.

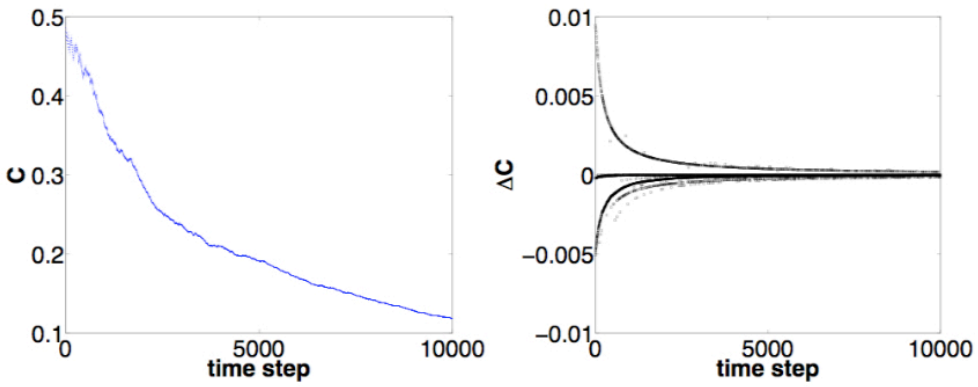

Figure 6.6. **Evolution of the average clustering coefficient C of a growing Barabasi-Albert network.** - In the left panel we show the evolution of the average clustering coefficient C at each time instance. In the right panel we show the change of average clustering coefficient ΔC. Note that to obtain this curve of the evolution of C, we only need to compute C for the initial network once, which requires $O(< k >^3 n)$ operations, and then adopt our update formula, which requires $O(< k >)$ operations per step; While if one uses direct computation, it would require $O(< k >^3 n)$ to compute each single step, and is highly inefficient. The structure in the right side graph is only observeable because we compute the change *at each time step* and would be obscured if we were to compute the change over larger time increments.

6.5.2 Evolution of Modularity

We start from an initial network with clear partition, constructed as follows: generate an empty graph of n vertices, and prescribe a partition of the set $\{1, 2, ..., n\}$ into two groups such that the group sizes are n_1, n_2. Randomly connect any pair of vertices in group 1 with probability p_1, and those in group 2 with probability p_2; then randomly connect a vertice in group 1 to a vertice in group 2 with probability $p_{between}$. Probability $p_{between}$ is chosen to be smaller than p_1 and p_2, creating a clear community structure. In our example, we choose $n = 1000$, with group 1 composed of nodes $\{1, ..., 500\}$, with the rest of the nodes forming group 2. We let $p_1 = p_2 = 0.2$ and $p_{between} = 0.05$. For the evolutionary process, we add random edges between the groups until the probability of connecting between groups is the same as the probability of connecting inside the groups (resulting in a completely random network in the end). In Fig. 6.7 we plot the adjacency matrix of the graph at three specific time instances. As more in-between edges are added, the original partition is less valid

(the block diagonal structure becomes more vague). Correspondingly, the evolution of the modularity measure is shown in Fig. 6.8.

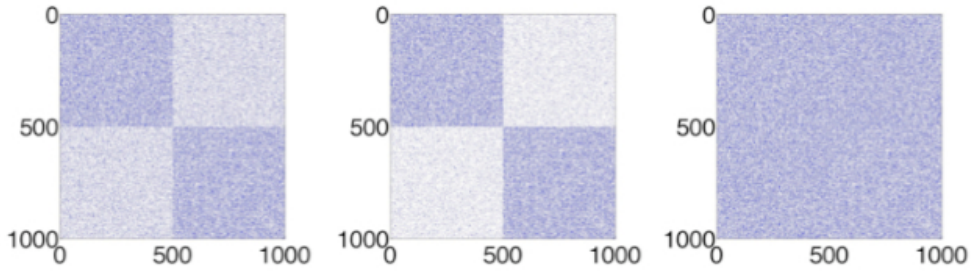

Figure 6.7. **Spy plot at three specific instances for the adjacency matrices of a random growing network.** - The left panel correponds to the initial network ($p_1 = p_2 = 0.2$ and $p_{between} = 0.05$), where there is a clear community structure. The middle panel corresponds to the time when $p_{between}$ reaches 0.1 where the community structure becomes less apparent. The right panel is the end of the growing process such that $p_{between} = 0.2$ and the network is totally random with no community structure.

6.6 Discussion and Open Problems

In this chapter we derive update formulae for important local network statistics (degree, clustering coefficient, assortativity coefficient, and modularity) as theoretical and computational tools for analyzing evolving networks. The update formulae are based on a singe edge or node updating. An arbitrary change to the graph structure can be viewed as a sequence of these unitary changes, with statistics updated by sequential applications of the formula we present in this paper. We also show several examples to illustrate the use of the updating scheme, allowing us to efficiently track the evolution of network statistics in situations where traditional methods to compute those statistics would be impractical.

The derivation of the update formula in this chapter requires that the statistics depend locally on network structure, for example, the update formula for the clustering coefficient only requires the knowledge of local information of the vertices that are going to be connected.

It becomes intriguing when the statistics we wish to update depends potentially

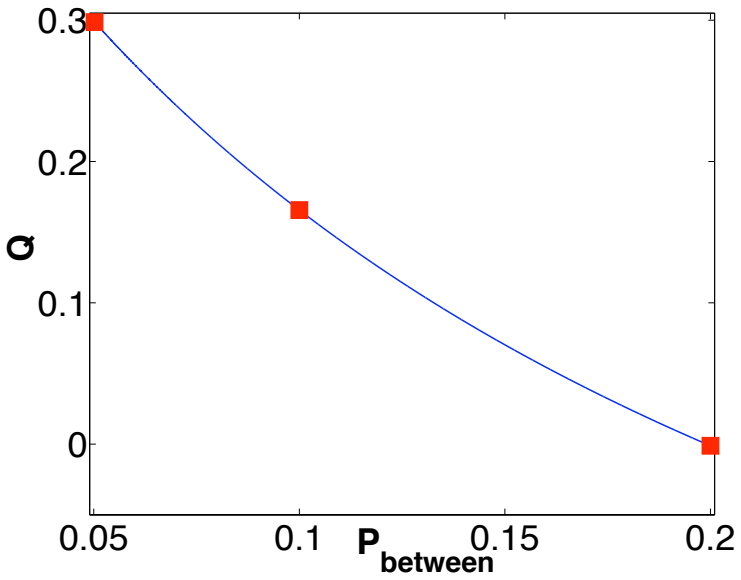

Figure 6.8. **Evolution of modularity Q for a random growing network.** - Three red squares correspond to the time instances that are shown in Fig. 6.7.

on *all* the edges of a graph, for example, shortest paths, and eigenvalues of an adjacency matrix. There exists related theory that can be used to bound the change of some global statistics when a the change in a graph is considered as a perturbation. For example, the perturbation to the spectra and eigenvectors (including the Fiedler vector) of the graph Laplacian upon adding or deleting a few edges in the graph may be obtained by results such as those in (Fie89) and (Dem97) based on Gershgorin theorem. However, in practice such bounds may be far from being sharp in the sense that the actual change happens to the statistics may be far from what the perturbation theory predicts. Thus, direct/naive use of the existing theory from linear algebra would provide little insight into the actual evolution of networks. In the next chapter (Chapter 7) we will make an attempt to tackle some of the difficulties. In particular, we will develop an update schema for efficiently computing path lengths in an evolving network. For spectrum related statistics, an exact update formula is impractical to find, since computing eigenvalues in general requires numerical procedures which suffer from small errors; instead we develop approximation formulae for the actual perturbation, and have shown by numerical examples that in many cases the developed approximation is satisfactory.

On the other hand, the examples of update schema of those local statistics shown

in this chapter motivate us to find a systematic way to update *all* graph statistics, in some sense. One possibility is to *classify* graph statistics into different orders, and derive framework for update graph statistics in general, including both local and global situation.

Chapter 7

Dynamics of Networks: Evolution of Global Statistics

In the previous chapter, and (SBBS09), updating schema for some important network statistics such as degree, clustering coefficient, assortativity, and modularity were reported, based on the use of local information from the network. However, efficient updating schema for statistics that rely on global information, such as the shortest paths from one vertex to another, or the spectra of the graph adjacency matrix or graph Laplacian can not be obtained by similar approach.

In this Chapter (Chapter 7) we introduce methods to update global statistics such as the path lengths and spectral radius of an adjacency matrix. The update formulae for all-pair shortest paths, although usually faster in practice, might, in the worst case scenario, cost as much as a re-computation. On the other hand, if we are only interested in update the *average path length*, instead of all the path lengths, an asymptotic better algorithm can be found, as long as the *diameter* (defined as the length of a longest path in a graph) of the graph scale *sub-linearly* as the number of nodes. In the case of spectrum related statistics, the best one can hope is approximation formulae, since the computation of eigenvalues and eigenvectors of a matrix would in general suffer from numerical error to begin with. Based on classical perturbation results, we have developed useful approximation formulae to

measure the impact of changing an arbitrary subgraph on the spectral radius of the adjacency matrix. Similar techniques can be easily adopted to develop formulae for other spectrum related statistics, such as the eigenvalues of the graph Laplacian.

7.1 Global Statistics: Exact Update vs. Approximation

7.1.1 Shortest Paths in Networks

Shortest paths in networks play important roles in many contexts. For a simple graph $G = (V, E)$, a *path* (Die06) is a sequence of nodes $(i_1, \ldots, i_{l+1})$ such that for each $k \in \{1, \ldots, l\}$, $(i_k, i_{k+1}) \in E$. The *length* of a path $(i_1, \ldots, i_{l+1})$ is simply l. A path between nodes s and t is a path such that in the above notation, $i_1 = s$ and $i_{l+1} = t$, or the other way around. A path $(i_1, \ldots, i_{l+1})$ from node s to t is called a *shortest path* if for any other path $(i'_1, \ldots, i'_{l'+1})$, its length $l' \geq l$; that is, a shortest path (connecting two certain nodes) is a path with minimal length. A shortest path between two given nodes is *unique* in trees (a tree is a loop-less graph), but not true in general. See Fig. 7.1 for illustration of these concepts.

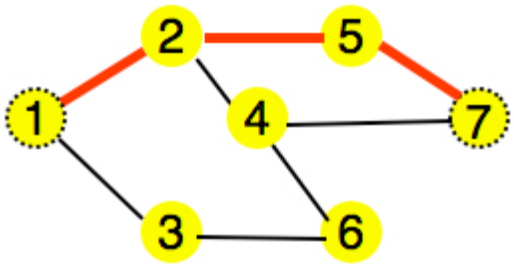

Figure 7.1. **Example of a shortest path in a small graph.** - For this small graph with 7 nodes, an example of a path can be $(2, 4, 7, 5, 2, 1, 3)$, of length 6. This path contain a loop $(2, 4, 7, 5, 2$, and thus can not be a shortest path. A shortest path connecting nodes 1 and 7 (dashed circles) is highlighted by bold lines, which is the path $(1, 2, 5, 7)$. Note that there are more than one shortest paths connecting nodes 1 and 7, namely, the paths $(1, 2, 5, 7)$ and $(1, 2, 4, 7)$, both of length 3.

Computing shortest paths between two nodes in a graph usually requires the full

knowledge of all the structure of the graph, since *potentially* any edge in the graph may be present in the shortest path. For simple graphs, a classical technique called *breadth-first-search (BFS)* can be used to compute shortest paths between a given node to all other nodes, with computational complexity (CLRS01; KT05) $O(m+n)$ where n, m are the number of nodes and edges in the graph, respectively. The shortest path lengths between all pairs of nodes can thus be obtained by performing BFS at all nodes in a graph, in $O(mn+n^2)$ operations. Details of BFS on such problems shall be reviewed in section 7.2.

As a global property, many real world networks studied so far are known to have low path lengths compared to their size, usually scale logarithmically as the number of nodes (AB02; New03; WS98); a popular technique for nonlinear dimensionality reduction constructs a graph whose shortest paths reveals the natural distances between data points on the underlying nonlinear manifold (TSL00), such technique can also be adopted for the modeling of chaotic attractors (Bol07); a novel approach to the graph isomorphism problem also relies on the computation of shortest paths (BBSb08); navigation on networks also usually relies on information about the shortest path structures (Kle00a; Kle00b). Furthermore, shortest paths are how they change when network changes is also relevant to statistical physics as a relevant problem in the study of percolation (LPC$^+$07).

A natural problem one may encounter, in practice, is that, when small changes are introduced to the graph, how much the current information can be used, to avoid the re-computation of shortest paths of the whole network, a costly task indeed. It is also crucial to have an efficient algorithm so that in modeling the evolution of networks one is able to compare the evolution of average path length of the model to real world data.

We, in this thesis, aim at developing efficient (although not necessarily optimal) algorithms to compute shortest paths in a similar manner as other update schema (see Section 7.2.2 for details). In the case of updating all-pair shortest path lengths, we illustrate by an example that in the worst case the actual change can indeed be $O(n^2)$ for a simple sparse graph and thus any algorithm that is able to update all path lengths has to scale no better than $O(n^2)$. However, if we only want to compute the average path length, there exists algorithm (Section 7.2.3) which runs in $O(d_m n)$

time, where d_m is the diameter of the graph under consideration (usually $d_m \sim \log(n)$ for small world networks).

7.1.2 Eigenvalues and Eigenvectors of Networks

Another important class of statistics related to large networks is spectrum-related statistics, such as the spectral radius of an adjacency matrix, denoted in the following simply by λ when no confusion occurs.

For an uncorrelated network [1], define $\mu \equiv \frac{\langle k^2 \rangle}{\langle k \rangle}$ where $\langle . \rangle$ denotes average. There is an interesting connection between λ and μ in large uncorrelated networks, as shown in (CLV03), that: $\lambda \approx \mu$ if $\mu > \bar{k}_{max} \log n$ where n is the number of vertices, and $\bar{k}_{max}$ is the averaged maximum degree over many realizations. The intuition is that: λ and μ are 'sort of' positively correlated, and the larger λ (or μ) is, the more 'connected' the network is.

Furthermore, it was shown in (ROH06b) that the critical coupling strength for a large network of coupled phase oscillators to achieve synchronization is proportional to $\frac{1}{\lambda}$; the threshold for epidemic in an uncorrelated network is $\frac{1}{\mu}$, and that of a correlated network is proportional to $\frac{1}{\lambda}$ where here the λ is for a *connectivity matrix* which relates to the original adjacency matrix (BP02b); regarding percolation and robustness of networks, the critical fraction of nodes that need to be removed to disintegrate the network is related to μ as: $p_c = 1 - \frac{1}{\mu - 1}$ (CEbH00). The list continues in various other problems across science and engineering, see (DGM08; Mac00; ROH06a; ROH07) and the references therein.

From another point of view, one of the most important and fundamental problems in complex networks, both from a theoretical and applicative viewpoint, is that of measuring centrality. Various measures of centrality have been proposed, to account for different quantitative properties of the underlying network. Examples are degree centrality, shortest path and random walk betweenness, clustering coefficient and eigenvector component (ER05; New03). One of the most common applications of such measures is probably the ranking of vertices of a network, in particular in the

[1] By uncorrelated we mean that the joint probability distribution of the node degrees are independent.

context of web search engines (Kle99; LM05). Many of these centrality measures are connected with the spectral radius of the adjacency matrix of the graph.

For a real world network that evolve in time, little is known about how its spectrum changes accordingly, and its consequences. Such measures would be important, as theory in problems such as synchronization of a complex interacting system usually described by large network of coupled oscillators may hold only in specific cases (PSBS06; SB04; SBR06) and not in general. How current theory on static networks can be extended to time dependent networks is an important problem. The difficulty comes, again, partly because computing eigenvalues of large matrices is already expensive (at least $O(n)$); when a network changes, tracking the exact change in some eigenvalue even just for one step requires considerable amount of time. Although an exact update is unlikely to be found, classical perturbation results for eigenvalues can be used in novel ways to *approximate* the change, for example, of the spectral radius of an adjacency matrix upon structural modification to the network even when the perturbation is not infinitesimal (see Section 7.3 for details). This type of technique can also be adopted for other statistics such as the eigenvalues of a graph Laplacian (MSN09).

7.2 Updating Path Lengths of Evolving Networks

Motivated by such problems discussed in the beginning of this chapter and various applications, we present an updating schema for shortest paths in a network. These schema update the distance and predecessor matrices of the corresponding graph upon elementary operations consisting of either addition or deletion of a single edge. Arbitrary changes can be decomposed into these elementary operations.

7.2.1 Breadth-First-Search

Compute a shortest path from one node to another is a classical problem in graph theory. A widely used technique for this problem in the case of simple graphs is called *breadth-first-search (BFS)* (CLRS01; KT05). Here we briefly review this approach.

Suppose that we want to find the shortest path lengths from a given node s to all other nodes in a given simple graph $G = (V, E)$. Denote the shortest path length between nodes i and j by $d(i, j)$ ($d(i, j)$ is also referred to as the *distance* between nodes i, j). By convention we let the $d(i, i) = 0$ for all i, and if there is no path between i and j, its distance is $d(i, j)$ is defined to be ∞.

It is obvious that the distance from s to all its neighbors is 1. To compute the distance to more nodes, one basically start to consider the neighbors of neighbors which do not connect directly to s, excluding s itself; those will have distance 2 to s. Then this process is repeated until all the nodes that can be reached from s have been explored, and by then the distances from s to all these nodes will be obtained, and the distances from s to all nodes that cannot be explored in this process are all ∞. If the graph is *connected*, then indeed all the nodes in the graph will be explored in the above process, otherwise the set of nodes defined above form a *component* of the graph.

Formally, this type of iterative search is called a breadth-first-search (BFS), and can be summarized more precisely in the following:

Given: $G = (V, E)$, an undirected, unweighted, and loop-less graph where $|V| = n$ and $|E| = m$

Goal: Find the shortest path lengths from a node s to all the other nodes and the corresponding paths.

Let: $d = [d_i]_{1\times n}$ and $\pi = [\pi_i]_{1\times n}$ be two vectors to represent the solution, where d_i represents the distance from node s to i, and π_i the predecessor of i on some shortest path from s.

Set: $b = [0, ..., 0]$ an $1 - by - n$ vector.

Set: A list $L = \{s\}$, $d_s = 0$, $\pi_s = 0$, and $b_s = 1$.

while $L \neq \emptyset$ **do**

  Set $C = L$; Set $L = \emptyset$.

  for all nodes $j \in C$ **do**

    Find all the neighbors of node j, forming a set $N(j)$.

    for all nodes $k \in N(j)$ **do**

      if $b_k = 0$ **then**

Set $d_k = d_j + 1$ and $\pi_k = j$, $b_k = 1$.
Add k to the list L.
end if
end for
end for
end while
for all i such that $b_i = 0$ **do**
Set $d_i = \infty$ and $\pi_i = \infty$.
end for

Note that the **while** loop runs through all edges in the graph at most once for each, and all other operations are all of order n, the algorithm thus can be shown to have the computational complexity $O(m+n)$ where n and m are the number of nodes $n = |V|$ and edges $m = |E|$ respectively. In the case of *sparse graphs* where $m = O(n)$, the complexity would simply be $O(n)$, i.e., linearly dependent on the number of nodes in the graph.

Fig. 7.2 illustrates the process of BFS by using a small graph as an example. Using the BFS algorithm described above, consider node 1 as the base node, one obtains the vectors $d = [0, 1, 1, 2, 2, 2, 3, 3, 4, 4]$ and $\pi = [0, 1, 1, 2, 2, 3, 5, 5, 8, 8]$ in this case. A BFS tree obtained by starting at a node s in a graph is referred to as a *BFS tree rooted at node s*. Note that a BFS tree rooted at a node need not be unique. In fact, for node 1 in the graph shown in Fig. 7.2, we could have replaced the edge $(2, 5)$ by $(3, 5)$ and obtain another BFS tree rooted at node 1. Nonetheless, the shortest path lengths between nodes are always unique, regardless the different possible choices of BFS trees.

To compute the path lengths between all pairs of nodes, one simply needs to perform the BFS algorithm at each individual node. Thus the computational cost for obtaining the shortest paths for all pairs of nodes becomes $O(mn + n^2)$, following the fact that a single BFS requires $O(m + n)$ operations, and there are n nodes in a graph.

We will use a matrix $D = [d_{ij}]$ to represent the distances between nodes, where d_{ij}

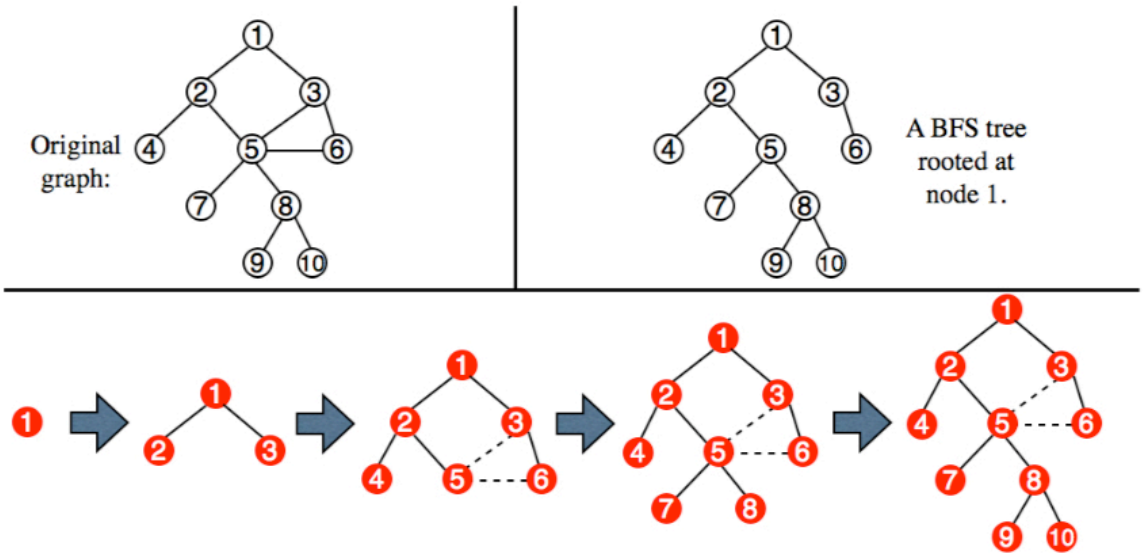

Figure 7.2. **Example of a BFS tree in a small graph.** - The upper left shows the example graph, where the upper right shows a BFS tree rooted at node 1. The lower panel shows, stage by stage, the process of BFS starting at node 1. The solid lines are edges included in the BFS tree, where the dashed lines indicate edges from the original graph which do not belong to the BFS tree.

equals the shortest path length between nodes i and j; and another matrix $\Pi = [\pi_{ij}]$ to store the predecessors, where π_{ij} denote the predecessor of node j on the shortest path from i in a BFS tree rooted at node i. So for example, the matrices D and Π for the graph shown in Fig. 7.2 are:

$$D = \begin{array}{c} \\ \mathbf{1} \\ \mathbf{2} \\ \mathbf{3} \\ \mathbf{4} \\ \mathbf{5} \\ \mathbf{6} \\ \mathbf{7} \\ \mathbf{8} \\ \mathbf{9} \\ \mathbf{10} \end{array} \begin{bmatrix} 0 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 4 & 4 \\ 1 & 0 & 2 & 1 & 1 & 2 & 2 & 2 & 3 & 3 \\ 1 & 2 & 0 & 3 & 2 & 1 & 3 & 3 & 4 & 4 \\ 2 & 1 & 3 & 0 & 2 & 3 & 3 & 3 & 4 & 4 \\ 2 & 1 & 2 & 2 & 0 & 1 & 1 & 1 & 2 & 2 \\ 2 & 2 & 1 & 3 & 1 & 0 & 2 & 2 & 3 & 3 \\ 3 & 2 & 3 & 3 & 1 & 2 & 0 & 2 & 3 & 3 \\ 3 & 2 & 3 & 3 & 1 & 2 & 2 & 0 & 1 & 1 \\ 4 & 3 & 4 & 4 & 2 & 3 & 3 & 1 & 0 & 2 \\ 4 & 3 & 4 & 4 & 2 & 3 & 3 & 1 & 2 & 0 \end{bmatrix}, \quad \Pi = \begin{array}{c} \\ \mathbf{1} \\ \mathbf{2} \\ \mathbf{3} \\ \mathbf{4} \\ \mathbf{5} \\ \mathbf{6} \\ \mathbf{7} \\ \mathbf{8} \\ \mathbf{9} \\ \mathbf{10} \end{array} \begin{bmatrix} 0 & 1 & 1 & 2 & 2 & 3 & 5 & 5 & 8 & 8 \\ 2 & 0 & 1 & 2 & 2 & 5 & 5 & 5 & 8 & 8 \\ 3 & 1 & 0 & 2 & 6 & 3 & 5 & 5 & 8 & 8 \\ 2 & 4 & 1 & 0 & 2 & 5 & 5 & 5 & 8 & 8 \\ 2 & 5 & 6 & 2 & 0 & 5 & 5 & 5 & 8 & 8 \\ 3 & 5 & 6 & 2 & 6 & 0 & 5 & 5 & 8 & 8 \\ 2 & 5 & 6 & 2 & 7 & 5 & 0 & 5 & 8 & 8 \\ 2 & 5 & 6 & 2 & 8 & 5 & 5 & 0 & 8 & 8 \\ 2 & 5 & 6 & 2 & 8 & 5 & 5 & 9 & 0 & 8 \\ 2 & 5 & 6 & 2 & 8 & 5 & 5 & 10 & 8 & 0 \end{bmatrix}, \tag{7.1}$$

where row indices have been put (in bold) on the left hand side of both matrices for convenience.

7.2.2 Updating All-Pair Shortest Paths

Motivated by problems and applications mentioned at the beginning of this chapter, we present an updating schema for efficiently computing shortest paths for graphs that change in time. The schema presented in this section update the distance and predecessor matrices of the corresponding graph upon elementary operations consisting of either addition or deletion of a single edge. Arbitrary changes can be decomposed into these elementary operations.

Consider a *sparse undirected graph* $G = (V, E)$ with no self-loops, where V, E are the vertex and edge sets respectively. Let $n = |V|, m = |E|$, then $m = O(n)$ since the graph is sparse. It is also assumed that the graph is connected throughout the section; for graphs that are not connected, one shall adopt our algorithms for each *connected component* of the graph. Since finding the connected component only requires $O(n)$ operations, the results obtained in this paper would not be affected. We will use A to represent the *adjacency matrix* associated with G, and $D = [d_{ij}]_{n\times n}$ be the *distance matrix*, where

$$d_{ij} \equiv \text{length of the shortest path between } i \text{ and } j; \tag{7.2}$$

And let $\Pi = [\pi_{ij}]_{n\times n}$ be the *predecessor matrix* which encodes the actual shortest paths between all pairs of nodes, where

$$\pi_{ij} \equiv \text{the predecessor of } j \text{ on some shortest path from } i. \tag{7.3}$$

By convention, set $d_{ii} \equiv 0$ and $\pi_{ii} \equiv 0$ for each i. We use (i, j) to denote the edge connecting vertices i and j, and $i \to j$ to denote the shortest path from i to j. There may be multiple shortest paths from i to j. We are only concerned with any one of them that has been discovered.

As discussed in the previous section, obtaining D and Π for a given unweighted graph requires $O(n^2)$ operations when the graph is sparse. Our goal is to develop an efficient algorithm that utilizes the current information of the network to efficiently compute the new distances and shortest paths upon small change to the network.

Adding or removing a leaf vertex

Suppose a new vertex is introduced, labeled as $n+1$, which only connects to a vertex p in the existing network ($1 \leq p \leq n$) with edge weight w. Such a vertex is usually referred to as a *leaf* in the graph. Since the path from vertex $n+1$ to any other vertex must go through its only neighbor p, and no other paths will use this edge, the new distance matrix can be easily obtained, as follows:

$$\widetilde{D} = \begin{bmatrix} D & \mathbf{d}_{:,p} + \mathbf{1} \\ \mathbf{d}_{p,:} + \mathbf{1}^{\mathrm{T}} & 0 \end{bmatrix}, \tag{7.4}$$

where $\mathbf{d}_{:,p}$ and $\mathbf{d}_{p,:}$ are the p^{th} column and row of D, respectively; and $\mathbf{1} \equiv [1, \ldots, 1]^{\mathrm{T}}$. It follows that for shortest paths,

$$\widetilde{\Pi} = \begin{bmatrix} \Pi & p\mathbf{1} \\ \pi_{p,:} + (n+1)\mathbf{e}_p^{\mathrm{T}} & 0 \end{bmatrix}, \tag{7.5}$$

where $\pi_{p,:}$ is the p^{th} row of Π, and $\mathbf{e}_p^{\mathrm{T}} \equiv [0, \ldots, 0, 1, 0, \ldots, 0]$ with the 1 at its p^{th} position. The above update requires $O(n)$ operations.

To delete a leaf vertex p, the inverse operations are performed, and simply delete the p^{th} row and column of D and Π.

Adding an edge between unconnected vertices

Suppose vertices p and q are not connected in G, and an edge is added between them, resulting in a new graph $\widetilde{G}$, with corresponding adjacency matrix $\widetilde{A}$, distance matrix $\widetilde{D}$, and predecessor matrix $\widetilde{\Pi}$.

Updating the distance and predecessor matrices consists of the following steps:

1. Identify the following sets:

$$\begin{aligned} V_p^+ &\equiv \{\, i \in V \mid d_{ip} + 1 < d_{iq} \}\,, \\ V_q^+ &\equiv \{ j \in V \mid d_{jq} + 1 < d_{jp} \}\,. \end{aligned} \tag{7.6}$$

   This step requires $O\left(n\right)$ operations. Note that it is always true that $p \in V_p^+, q \in V_q^+$.

It can be shown [2] that the distance between any pair of vertices (i, j) may change only if $(i, j) \in (V_p^+ \times V_q^+) \cup (V_q^+ \times V_p^+)$. An important consequence is that *any new shortest path only consists of vertices in* $V_p^+ \cup V_q^+$.

2. Update the shortest paths.
   Let $i \in V_p^+$. Update the i^{th} row of D, Π according to:

$$\begin{aligned} \widetilde{\mathbf{d}}_{i,V_q^+} &= d_{ip} + 1 + \mathbf{d}_{q,V_q^+}, \\ \widetilde{\pi}_{i,q} &= p, \\ \widetilde{\pi}_{i,V_q^+ - \{q\}} &= \pi_{q,V_q^+ - \{q\}}. \end{aligned} \tag{7.7}$$

   Similarly we can update the rows for which $j \in V_q^+$: simply replace i by j, p by q, and q by p in Eq. (7.7).

The evaluation of Eq. (7.7) runs through all pairs of vertices $(i, j) \in (V_p^+ \times V_q^+) \cup (V_q^+ \times V_p^+)$. Let $n_+ = \frac{1}{2}\left(\left|V_p^+\right| + \left|V_q^+\right|\right)$, then this step requires $O\left(n_+^2\right)$ operations. Hence, the total operations for the update is $O\left(n + n_+^2\right)$.

Fig. 7.3 shows a simple example of the sets defined by Eq. (7.6). Fig. 7.4 provides simulations of the size of n_+.

This update schema, although not necessarily optimal, might be the best one can hope, since there are examples where n_+ actually scales as n, and the number of distances change in the graph is actually of $O(n^2)$. Such an example can be simply constructed, see Fig. 7.5.

Removing an existing edge

Suppose (p, q) is an edge in G. The graph after the deletion of edge (p, q) is denoted as $\widehat{G}$, with associated adjacency matrix $\widehat{A}$, distance matrix $\widehat{D}$, and predecessor matrix $\widehat{\Pi}$.

[2] The proof is based on extensive use of the triangle inequality property of distances between vertices, and the fact that any sub-path of a shortest path is itself a shortest path. The details are boring and thus omitted from the thesis.

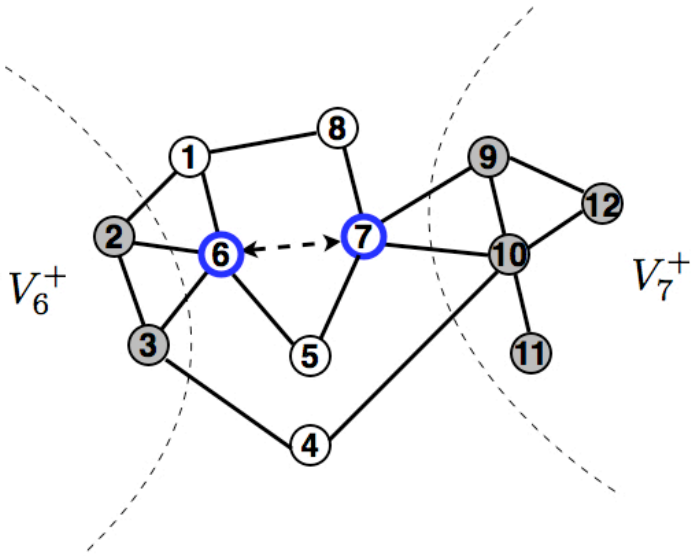

Figure 7.3. **The sets V^+ upon edge addition.** - Solid lines are unchanged edges in the original graph (unweighted), while the dashed line with arrowheads connecting vertices $p = 6$ and $q = 7$ is the edge added. $V_p^+ = \{2, 3\}$, and $V_q^+ = \{9, 10, 11, 12\}$ (shaded circles) by Eq. (7.6), which denote the vertices that use the newly added edge to construct shortest paths to q and p respectively.

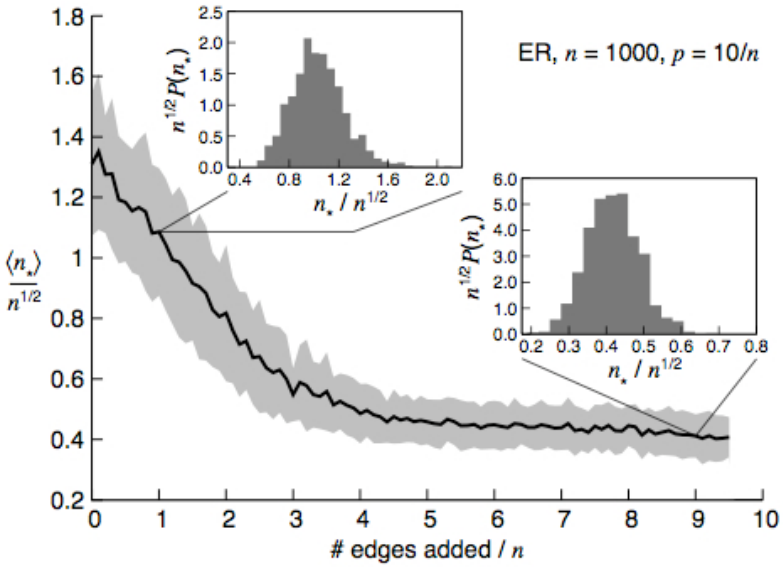

Figure 7.4. **The size of n_+ under random edge addition.** - Shown are simulations for Erdos-Renyi (ER) graphs, averaged over 100 runs, where random unconnected pairs of vertices were connected and n_+ was computed (shaded regions indicate $\pm$ one standard deviation). Insets show the full distribution at the annotated points. As the edge density increases, shortcuts become less important, and n_+ decreases.

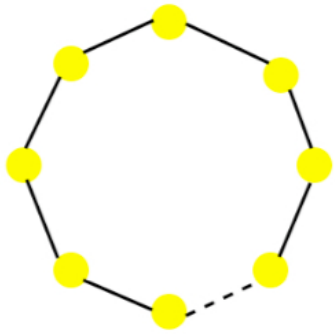

Figure 7.5. **Worst case scenario of the update schema upon addition of an edge.** - Consider a ring graph of n nodes with one edge missing, as illustrated in the picture. When this edge (dashed line) is added to the graph, the sets V_p^+ and V_q^+ are both of size $\frac{n}{2}$; on the other hand, the actual change in the distance matrix is also of order n^2. Trivial as it seems, this example shows that, update is not necessarily better than re-compute, at least in the asymptotic case.

1. Identify the following sets (in $O(n)$ operations):

$$\begin{aligned} V_p^- &\equiv \{ i \in V \mid d_{ip} + 1 = d_{iq} \}, \\ V_q^- &\equiv \{ j \in V \mid d_{jq} + 1 = d_{jp} \}. \end{aligned} \tag{7.8}$$

   Similarly as the case of addition, it can be shown that the distance between any pair of vertices (i, j) may change only if $(i, j) \in (V_p^- \times V_q^-) \cup (V_q^- \times V_p^-)$, and any new shortest path only consists of vertices in $V_p^- \cup V_q^-$.

2. Update the shortest paths.

   Denote the subgraph of G *induced by* $V_p^- \cup V_q^-$ by G^-, and let $n_- = \frac{1}{2}\left(\left|V_p^-\right| + \left|V_q^-\right|\right)$. We obtain the shortest paths between vertices in this subgraph G^- (i.e., distance and predecessors for this induced subgraph) by standard BFS algorithm. This step thus costs $O\left(n_-^2\right)$ operations.

Thus, the number operations needed for the updates in the case of removal is $O\left(n_-^2\right)$.

Fig. 7.6 shows a simple example of sets (7.8).

Again, consider the case illustrated by Fig. 7.5, in the worst case, $n_- \sim n$, and update might require $O(n^2)$ operations, although the leading constant could be smaller than that of performing a re-compuation.

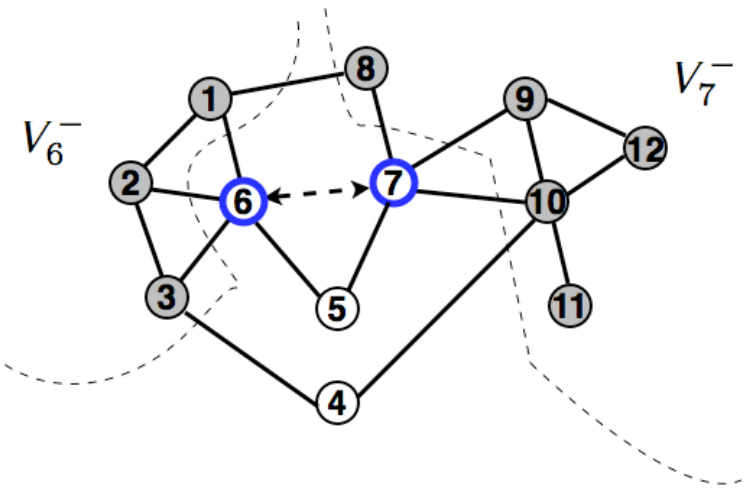

Figure 7.6. **The sets V^- upon edge removal.** - Solid lines are unchanged edges in the original graph (unweighted), while the dashed line with arrowheads connecting vertices $p = 6$ and $q = 7$ is the edge removed. $V_p^- = \{1, 2, 3\}$, and $V_q^- = \{8, 9, 10, 11, 12\}$ (shaded circles) by Eq. (7.8), which denote the vertices that could have used the removed edge for its shortest paths to q and p respectively.

7.2.3 Updating Average Path Length

In the following we discuss how to update simply the *average path length* without going into the details of updating al the shortest path lengths. This may be achieved by introducing a new count during the BFS process. This count will be denoted by a matrix $C = [c_{ij}]_{n\times n}$, where

$$c_{ij} \equiv \#\text{ offsprings of node } j \text{ in the BFS tree rooted at node } i. \tag{7.9}$$

When an edge (p, q) is added to a graph, for each i, there are three possibilities (analogous to the definition of sets V^+ and V^- described in the last subsection):

1. $|d_{ip} - d_{iq}| \leq 1$

   In this case the addition of edge (p, q) will not affect the distance from i to the other nodes.

2. $d_{ip} + 2 \leq d_{iq}$

   In this case, in the BFS tree rooted at i, node q and all its offsprings will go through the newly added edge (p, q) in their shortest paths to the root i. The

number of these nodes is precisely $c_{iq} + 1$, by definition. The change of the summation of distances from i to all other nodes follows:

$$\sum_j d_{ij} \to \sum_j d_{ij} + (d_{ip} - d_{iq})(c_{iq} + 1). \tag{7.10}$$

Also, the $i-th$ row of C will change. The update of this row can be performed by tracking the shortest paths from nodes p, q to the root i:
Let $(p, j_1, \ldots, j_x, i)$ and $(q, k_1, \ldots, k_y, i)$ be the shortest paths from p and q to i in the original graph, respectively. These paths can be obtained easily from reading the $i - th$ row of the matrix Π. We update row i of C as follows:

$$\begin{aligned} &\text{For each } l \in \{p, j_1, \ldots, j_x\} : c_{il} \to c_{il} + c_{iq} + 1; \\ &\text{For each } l \in \{k_1, \ldots, k_y\} : c_{il} \to c_{il} - c_{iq} - 1. \end{aligned} \tag{7.11}$$

These updates require at most $O(d_m)$ operations, where d_m is the maximum shortest path length, also known as the *diameter* of the graph.
Meanwhile, the update for Π matrix only requires changing one entry $\pi(i, q) \to p$.

3. $d_{iq} + 2 \leq d_{ip}$ This case is similar as the case where $|d_{ip} - d_{iq}| \leq 1$.

To summarize, once we have the matrices C and Π, and distances d_{ip}, d_{iq} for all i, the average path length $\langle d_{ij} \rangle$ can be simply updated without updating the whole matrix D. For each i, update the $i-th$ row of C and Π only requires $O(d_m)$ operations where d_m is the diameter of the graph; thus, update the whole C and Π matrices in the worst case requires $O(d_m n)$ operations. On the other hand, computing d_{ip}, d_{iq} for all i needs $O(n)$ operations.

Thus, we can conclude that the above approach updates the average path length in $O(d_m n)$ operations. In cases where $d_m \sim \log n$, we have an $O(n \log n)$ update. Fig. 7.7 illustrates the effect of adding an edge on the matrix C. The application of this approach to physical problems can be found in (SBB+09).

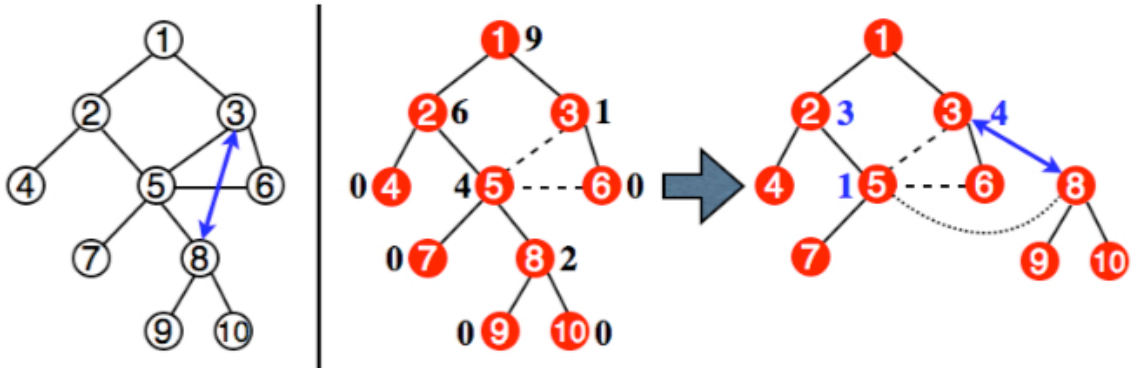

Figure 7.7. **Updating the number of offsprings in a BFS tree.** - Consider the graph shown in the left panel where black lines are the edges. Now suppose that we add an edge between nodes 3 and 8, denoted by a blue double arrow. The two pictures in the right panel shows the corresponding change of the BFS tree rooted at node 1, where dashed lines are the edges in the graph that are not present in the BFS tree. Bold numbers beside the nodes are the number of offsprings in this tree. For the new BFS tree the number is only shown where changes occur.

7.2.4 Application to Other Global Statistics

Eccentricity and Diameter

The eccentricity ε_i of a vertex i is simply:

$$\varepsilon_i \equiv \max_j [d_{ij}] \,. \tag{7.12}$$

To update these, simply maintain $\{\varepsilon_i\}$ for all vertices and, when $d_{ij} \to \widetilde{d}_{ij}$ (or $\widehat{d}_{ij}$), check if $\widetilde{d}_{ij} > \varepsilon_i$, and replace ε_i accordingly.

Likewise, the diameter $\mathcal{D}$ of the graph is the largest eccentricity:

$$\mathcal{D} \equiv \max_i [\varepsilon_i] \,. \tag{7.13}$$

This can be updated similarly. For each update $d_{ij} \to \widetilde{d}_{ij}$, if $\widetilde{d}_{ij} > \varepsilon_i$, check if $\widetilde{d}_{ij} > \mathcal{D}$ and update $\mathcal{D}$ accordingly.

Network Portraits

Define a network portrait (BBSb08) as the matrix B with elements:

$$b_{l,k} \equiv \#\text{ of starting nodes with } k \text{ nodes at a distance } l. \tag{7.14}$$

It has been shown (BBSb08) that these portraits encode a great deal of information about the network. Unfortunately, they are expensive to calculate, equivalent to computing the entire distance matrix D. If the network is evolving, generating a new portrait after each alteration will continually require $O(n^2)$ additional computations. This prohibitive cost can be mitigated using the shortest paths updating schemes presented here.

To update the portrait, define an auxiliary data structure, a "histogram vector" $\mathbf{s}_i$:

$$s_i(d) \equiv \#\text{ of nodes at distance } d \text{ from node } i, \tag{7.15}$$

which can be constructed initially alongside the original distance matrix. Upon each distance update, the corresponding $\mathbf{s}$ can be updated at $O(1)$ cost:

$$d_{ij} \to \widetilde{d}_{ij} \;\Rightarrow \begin{cases} s_i(d_{ij}) \to s_i(d_{ij}) - 1, \\ s_i(\widetilde{d}_{ij}) \to s_i(\widetilde{d}_{ij}) + 1; \end{cases} \tag{7.16}$$

and likewise for $d_{ij} \to \widehat{d}_{ij}$. The simultaneous update for d_{ji} follows from symmetry (for undirected networks).

Finally, B can be updated using $\mathbf{s}$:

$$d_{ij} \to \widetilde{d}_{ij} \;\Rightarrow \begin{cases} b_{d_{ij},s_i(d_{ij})} \to b_{d_{ij},s_i(d_{ij})} - 1, \\ b_{\widetilde{d}_{ij},s_i(\widetilde{d}_{ij})} \to b_{\widetilde{d}_{ij},s_i(\widetilde{d}_{ij})} + 1; \end{cases} \tag{7.17}$$

where $s_i(d_{ij})$ is the value *before* the update. Thus we can take advantage of the shortest paths updating schema to update the portrait at $O(1)$ cost, while using, e.g., only $O(n \ln n)$ additional storage (for a small-world network).

7.3 Approximating Spectrum Perturbations

In this section we discuss how to approximate the *change* of spectrum of a graph when small changes occur. For convenience of presentation we have assumed that the graphs under consideration are undirected and unweighted. Generalization can be found in (MSN09).

7.3.1 Defining the Spectral Impact

Let $G = (V, E)$ be a simple graph (undirected and unweighted and without self-loops), where V is the set of vertices, and E the set of edges. For convenience, the vertices in V can be labelled with integers $1, \ldots, n$.

Let $A \in \mathbb{R}^{n \times n}$ be the adjacency matrix of a simple graph (not necessarily simple), where its entry a_{ij} is defined to be 1 if there is an edge connecting nodes i and j, and 0 otherwise. Thus A is non-negative. If it is also *irreducible* (LT85), then the *Perron-Frobenius* theorem (LT85; Mac00) can be used to show that its largest eigenvalue λ is simple and positive, and there exist a positive eigenvector associated with it. In this case, denote the normalized eigenvector by v, so that

$$Av = \lambda v, \quad \|v\|_2 = 1. \tag{7.18}$$

In general, the condition of non-negativity is equivalent to requiring the weights of the edges of the underlying graph to be non-negative; on the other hand, the condition of irreducibility corresponds to the graph being strongly connected. In this section, the graphs under consideration all share these properties.

In (ROH06a) the *dynamical importance* I_{ij} of an edge from i to j was defined as the amount of relative decrease the removal of this edge causes on λ, i.e., if after removing such an edge λ becomes $\lambda - \Delta\lambda_{ij}$, then $I_{ij} \equiv \Delta\lambda_{ij}/\lambda$. Hence the dynamical importance of an edge quantitatively captures the effect that the removal of such an edge has on the largest eigenvalue of the graph adjacency matrix. An analogous definition was introduced for the dynamical importance of vertices, where I_k is the importance of node k. Approximations of these dynamical importances, based on perturbation techniques, are given in (ROH06a); in case of simple graphs, I_{ij} and I_k can be approximated, respectively, by

$$\bar{I}_{ij} = \frac{v_i v_j}{\lambda} \tag{7.19}$$

and

$$\bar{I}_k = v_k^2, \tag{7.20}$$

where the subscript i indicates the i-th component of the vector.

In this thesis, the notion of dynamical importance is extended to measure the *spectral impact (SI)* upon the removal of an arbitrary subgraph, and of the addition

of an arbitrary set of links. Formulae for approximating the SI will be given in the case of simple graphs. The readers are referred to the paper (MSN09) for general cases, including similar approach for the graph Laplacian.

The spectral impact is defined as [3]

$$I_B \equiv \frac{\lambda|_{A+B} - \lambda|_A}{\lambda|_A}, \tag{7.21}$$

where A denotes the adjacency matrix of the original graph, $\lambda|_A$ is the corresponding spectral radius, while B is the adjacency matrix of a subgraph of K_n (complete graph of n nodes), with weighted edges. Note that in the case of removing a subgraph of A, the entries of B will be non-positive.

7.3.2 Classical Perturbation Results and Approximation in Practice

Classical perturbation results for eigenvalues can be found, for example, in (Dem97; GV96; LT85; Wil65). Here we review some of the main results by Wilkinson (Wil65).

Let A and C be two $n-by-n$ real symmetric matrices satisfying $|a_{ij}| < 1, |c_{ij}| < 1$ for all i, j. Let λ_1 be a simple eigenvalue of A. The problem is to find the perturbation to this eigenvalue upon small change in A. By expanding the *characteristic equation* in terms of ϵ, the corresponding eigenvalue of the matrix $A + \epsilon C$ may be written as, for sufficiently small ϵ:

$$\lambda_1(\epsilon) = \lambda_1 + k_1\epsilon + k_2\epsilon^2 + \ldots, \tag{7.22}$$

where the coefficient k_1 can be obtained from:

$$k_1 = v_1^T C v_1, \tag{7.23}$$

where v_1 is the normalized eigenvector associated with λ_1. For k_2, *all* the eigenvalues and eigenvectors of A comes into play, even in the symmetric case:

$$k_2 = \sum_{i=2}^{n} \frac{(v_i^T C v_1)^2}{\lambda_1 - \lambda_i}. \tag{7.24}$$

[3] Here positive values correspond to an increase in λ, and negative values correspond to a decrease.

Since the second order term $k_2\epsilon^2$ involves the computation of all eigenvalues and eigenvectors of a matrix, it can be of little help in practice.

Rewrite Eq. (7.22) in a Taylor like expansion:

$$\begin{aligned}\lambda_1(\epsilon) &= \lambda_1 + \Delta\lambda = \lambda_1 + \Delta_1 + \Delta_2 + \dots \\ &= \lambda_1 + \epsilon\, \lambda'|_A + \frac{\epsilon^2}{2}\, \lambda''|_A + \dots,\end{aligned} \tag{7.25}$$

where by the classical results, we have the first order approximation:

$$\Delta\lambda \approx \Delta_1 = \epsilon\, \lambda'|_A = \epsilon v_1^T C v_1, \tag{7.26}$$

and for more accuracy, a second order approximation:

$$\begin{aligned}\Delta\lambda \approx \Delta_1 + \Delta_2 &= \epsilon\, \lambda'|_A + \frac{\epsilon^2}{2}\, \lambda''|_A \\ &= \epsilon\Big[v_1^T C v_1\Big] + \frac{\epsilon^2}{2}\Big[2\sum_{i=2}^{n} \frac{(v_i^T C v_1)^2}{\lambda_1 - \lambda_i}\Big].\end{aligned} \tag{7.27}$$

Since computing the second derivative term using known perturbation results would require knowledge of all the eigenvalues and eigenvectors of A (Wil65), as mentioned before, the second order approximation formulae is impractical. Hence, a further approximation is introduced, as

$$\lambda''|_A \approx \frac{\lambda'|_{A+\epsilon C} - \lambda'|_A}{\epsilon}. \tag{7.28}$$

The term $\lambda'|_{A+\epsilon C}$ takes into account the change in the eigenvector from v to $v + \Delta v$. We propose to estimate $v + \Delta v$ by means of one iteration of the power method. Here, the assumption is that v and $v + \Delta v$ are close to be parallel, hence a single iteration of the power method is satisfactory. The spectral gap of $A + \epsilon C$ is not critical for this application. Starting from the unperturbed eigenvectors, as

$$\begin{aligned}v + \Delta v &\approx \frac{(A + \epsilon C)v}{\|(A + \epsilon C)v\|_2} = \frac{\lambda v + \epsilon C v}{\|\lambda v + \epsilon C v\|_2} = \\ &= \frac{\lambda v + \epsilon C v}{\sqrt{\lambda^2 v^T v + 2\epsilon\lambda v^T C v + \epsilon^2 v^T C^T C v}} \\ &\approx \frac{\lambda v + \epsilon C v}{\lambda},\end{aligned} \tag{7.29}$$

where ϵ terms in the denominator have been neglected. Thus

$$\Delta v \approx \frac{\epsilon}{\lambda} C v. \tag{7.30}$$

Therefore, the derivative $\lambda'|_{A+\epsilon C}$ can be approximated as

$$\begin{aligned}\lambda'|_{A+\epsilon C} &= (v+\Delta v)^T C\,(v+\Delta v)\\ &\approx v^T C v + v^T C \Delta v + \Delta v^T C v \approx\\ &\approx v^T C v + \frac{2\epsilon}{\lambda} v^T C C v, \end{aligned} \tag{7.31}$$

having neglected terms containing Δv in the denominator, and the product $\Delta v^T C \Delta v$ in the numerator. Thus, an improved approximation for $\Delta\lambda$ is given by

$$\Delta\lambda \approx \Delta_1 + \Delta_2 \approx v^T \epsilon C v + \frac{1}{\lambda} v^T \epsilon^2 C^2 v. \tag{7.32}$$

7.3.3 Spectral Impact of Nodes, Edges, and General Subgraphs

Changing an Arbitrary Subgraph

In most interesting problems in the context of networks, discrete changes take place, instead of infinitesimal perturbations. In such cases, $A \to A + B$ and the entries of B are finite. Furthermore, the non-zero entries of B are usually of the same order of magnitude as the non-zero entries of A. If, however, the modifications introduced to the network are limited, then $\|B\|_2 \ll \|A\|_2$, and Eqs. (7.26) and (7.26) would be valid, with ϵC replaced by B. The other approximations we have made are likely to be valid also if the modification is small, and this is supported by the fact that Eq. (7.32) improves significantly over Eq. (7.26) for the example networks we will see below.

For a general change in the adjacency matrix from A to $A + B$, we thus have:

$$\Delta\lambda \approx \Delta_1 = v^T B v, \tag{7.33}$$

while the second order one is recast as

$$\Delta\lambda \approx \Delta_1 + \Delta_2 \approx v^T B v + \frac{1}{\lambda} v^T B^2 v. \tag{7.34}$$

Correspondingly, the SI I_B defined in Eq. (7.21) has first order approximation

$$\hat{I}_B = \frac{1}{\lambda} v^T B v, \tag{7.35}$$

and second order one

$$\hat{\hat{I}}_B = \frac{v^T B(v + Bv/\lambda)}{\lambda}. \tag{7.36}$$

Equation (7.35) is linear in B: therefore, the change can be decomposed into the sum of elementary changes, as $B = \sum_i B_i$, where B_i can represent, for example, a modification of a single edge. The corresponding first order approximation for the SI is obtained from the individual contributions, as $\hat{I}_B = \sum_i \hat{I}_{B_i}$. On the other hand, Eq. (7.36) has a linear and a quadratic dependence on B and linear superposition cannot in general be used. However, if all the products of elementary changes $B_i B_j$ are zero matrices (for example, if B_i's represent the disconnected components of the subgraph), then $\hat{\hat{I}}_B = \sum_i \hat{\hat{I}}_{B_i}$.

Removing an Edge

From Eqs. (7.35) and (7.36), the SI upon the removal of an edge (i, j) can be simply approximated at first and second order as:

$$\hat{I}_{B_{ij}} = -\frac{2v_i v_j}{\lambda}, \tag{7.37}$$

and

$$\hat{I}_{B_{ij}} = -\Big(\frac{2v_i v_j}{\lambda} + \frac{v_i^2 + v_j^2}{\lambda^2}\Big), \tag{7.38}$$

based on the fact that $B_i j$ is a matrix with entry being nonzero only at positions (i, j) and (j, i).

Removing a Node

Removing a node, indexed by k, in a simple graph corresponds to removing all edges touching it. In this case, the first order approximation reads

$$\hat{I}_k = -2v_k^2, \tag{7.39}$$

while the second order one yields

$$\hat{\hat{I}}_k = \left(-1 + \frac{d_k}{\lambda^2}\right) v_k^2, \tag{7.40}$$

where $d_k \equiv \sum_{i=1}^{n} a_{ki}$ is the *in-degree* of vertex k.

The last two equations show well that, in the case of removing a node, first and second order approximations yield rather different results, with Eq. (7.39) estimating an SI more than double than the one of Eq. (7.40). This difference comes from the

fact that, in the case of removing node k, the k-th component of the new dominant eigenvector becomes zero, regardless of its previous value, thus the change in v is not negligible.

7.3.4 Numerical Results

The accuracy of various approximations are assessed using both synthetic and real world graphs, shown in the following. As an example of artificial networks, the Erdos-Renyi random graph (Bol01) is used, with $n = 1000$ nodes and probability of connection $p = 0.01$. The particular realization used is labelled, for convenience, $\mathcal{G}_1$, and has 5004 undirected links, without self-loops. The largest degree is $d_{max} = 20$, the minimum is $d_{min} = 2$, the average $d_{mean} = 10.01$. The largest (in magnitude) eigenvalues of the corresponding adjacency matrix are computed to be $\lambda = 11.0741$, $\lambda_2 = -6.53518$ and $\lambda_3 = 6.50196$. The approximation to SI is shown in Fig. 7.8.

Note that for both the first and second order estimates the plots are almost monotonic, indicating that the ranking of nodes, edges, and triangles defined by the SI are accurately estimated by these formulae.

The approximation of SI is analyzed also on three real-world networks, referred to, for convenience, as $\mathcal{G}_2$, $\mathcal{G}_3$ and $\mathcal{G}_4$. Basic properties of such networks are reported in Table 7.1, along with the pertinent references. $\mathcal{G}_2$ is a biological example, $\mathcal{G}_3$ is a social interaction case, while $\mathcal{G}_4$ can be regarded as an instance having both an engineering and social character. Figure 7.9 shows – for these three networks – the eigenvector components, the real and approximated SI for removing an edge and removing a node. The eigenvector components are shown in order of increasing magnitude; in several cases, the smallest components appear to be rather small and fall below the axis limit on the figure. The presence of components in the dominant eigenvector spanning several order of magnitudes amounts to large discrepancies in the importance of edges, for example. When the removal of edges is analyzed, both the first and second order approximations for the SI are satisfactory. On the other hand, if removal of nodes is considered, the second order formula of Eq. (7.40), containing a correction for the degree of the node, yield results more accurate than Eq. (7.20). In this case, the SI is as large as -7%.

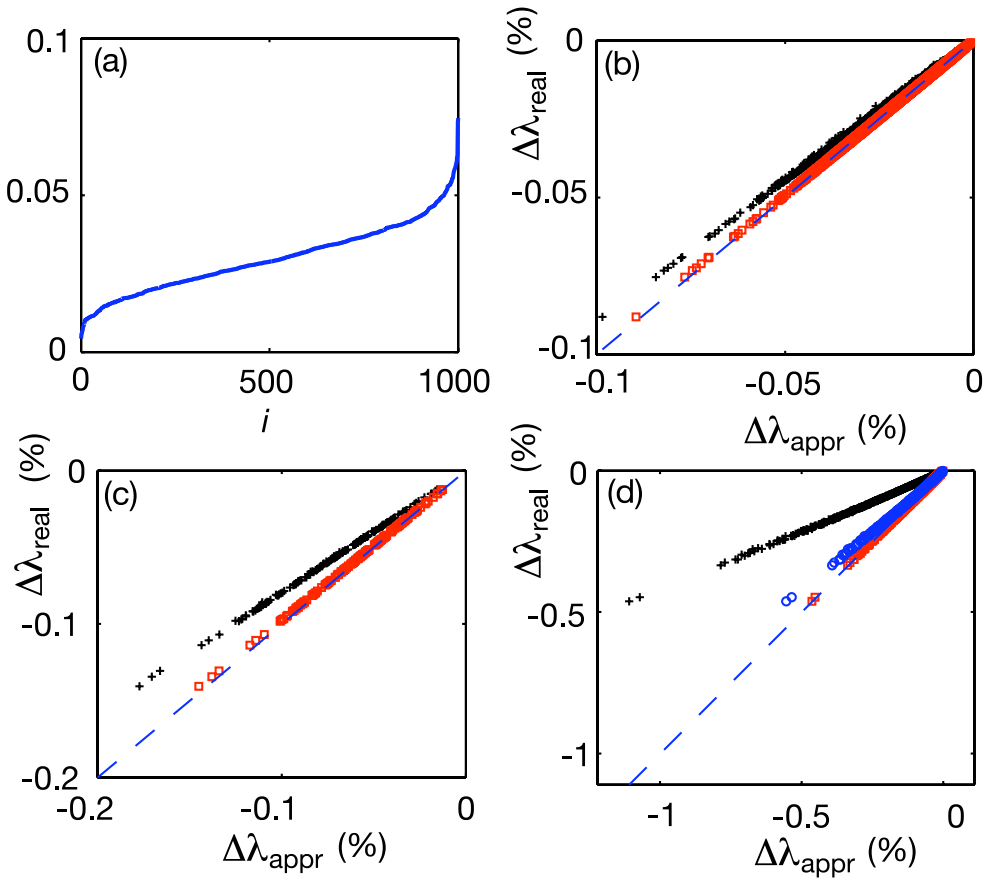

Figure 7.8. **Results for approximating the SI of an Erdos-Renyi network $\mathcal{G}_1$.** - (a) Eigenvector components sorted in increasing order. The true SI is plotted against its approximation (both in percentage) for the removal of (b) edges, (c) triangles, and (d) nodes. The black plus symbols correspond to the first order approximation (7.35), the red squares to the improved approximation (7.36), and the blue circles to the approximation in Ref. (ROH06a).

Table 7.1. Examples of real world networks.

$\mathcal{G}_2$, yeast protein interaction network (BZC$^+$03)
2361 vertices, 13828 arcs
$d_{min} = 1$, $d_{mean} = 5.86$, $d_{max} = 65$
$\lambda = 19.4861$, $\lambda_2 = 16.1340$, $\lambda_3 = 14.3339$
$\mathcal{G}_3$, network of e-mail interchanges (GDD$^+$03)
1133 vertices, 5451 edges
$d_{min} = 1$, $d_{mean} = 9.62$, $d_{max} = 71$
$\lambda = 41.4940$, $\lambda_2 = 33.9272$, $\lambda_3 = 30.0687$
$\mathcal{G}_4$, US power grid (Tsa) 1133 vertices, 6594 edges
$d_{min} = 1$, $d_{mean} = 2.67$, $d_{max} = 19$
$\lambda = 7.4831$, $\lambda_2 = 6.6092$, $\lambda_3 = 5.5728$

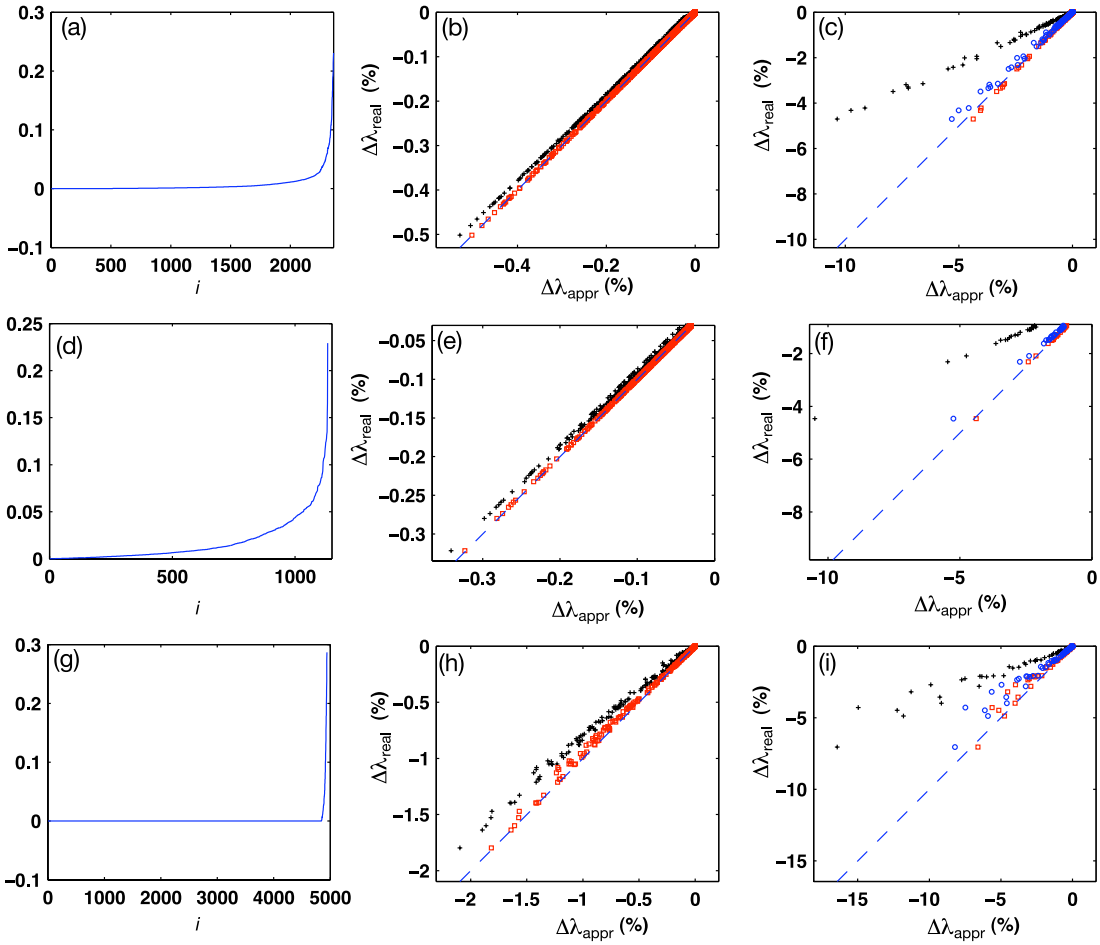

Figure 7.9. **Results for approximating the SI of three examples of real-world networks.** - Shown are the approximation of SI to networks: $\mathcal{G}_2$ (upper row), $\mathcal{G}_3$ (middle row) and $\mathcal{G}_4$ (lower row) in Table 7.1. The left column [(a), (d), and (g)] shows the eigenvector components sorted in increasing order. The middle column [(b), (e), and (h)] shows the true SI vs approximated SI for the removal of edges, while the right column [(c), (f), and (i)] shows the same plot for the removal of nodes. The meaning of the symbols is the same as in Fig. 7.8.

7.3.5 Related Problems for Future Work

In this section, we have defined the spectral impact of an arbitrary addition/removal of links as the resulting relative change in the largest eigenvalue of the adjacency matrix. Based on the standard perturbation method for eigenvalue problems and further approximation for the second order term, we obtained an improved first-order approximation formulae for the spectral impact. Using Erdos-Renyi random graph, as well as examples of large complex networks from biological, social, and technological applications, we confirmed the accuracy of the formulae for the addition and/or removal of nodes, links, triangles, and another network motif.

Our approximation works in certain class of networks. How different network topology affect the accuracy of this approximation? What does this mean in practice? Another related question is, how 'small' the perturbation B should be, and what is the appropriate measure for 'small', in a more rigorous sense?

In view of accident control in large networks, when sudden change (such as the removal of multiple edges or nodes in different places) happens and it is impossible to fix those immediately, there may exist certain strategy to change other parts of the network instead. What is the optimal way of doing that when λ is involved?

Also, designing networks that satisfy certain spectrum properties is an important problem for studying dynamics on networks (HSS06; HS08; SNb08). Our formulation may be used to develop gradient descent like algorithms to target the desired spectrum, numerically.

The ranking of subgraphs in terms of spectrum properties. Our formulation can be used directly to define an order topology on the set of subgraphs, given any graph and indicated eigenvalue (not necessarily the largest one). How does this order topology interplay with the other network properties? Is there a difference between different types of networks? Our improved formula suggests a nonlinear effect: i.e., the importance of two nodes/edges is not simply the addition of their respective importance values. How does this nonlinear effect correlate with the topological structure, such as communities in networks?

Chapter 8

Information of Networks: Graph Compression by Exploiting Symmetry

8.1 Introduction

8.1.1 Motivation

As mentioned before in this thesis, large complex networks have been studied extensively in the past ten years in the fields of mathematics, physics, computer science, biology, sociology, etc (BA99; AB02; New03; WS98; Wat99). Various networks are used to model and analyze real world objects and their interactions with each other. For example, in sociology, airports and airflights that connect them can be represented by a network (Paj); in biology, yeast reactions is also modeled by network (BZC$^+$03); etc.

The mathematical terminology for a network is conveniently described in the language of graph theory (Wes00). A common encoding of graphs uses an adjacency matrix, or an edge list, when the adjacency matrix is sparse. However, even for a

large network the edge list contains a large information storage. In the case that some important network is transfered frequently between computers, it will save time and cost if there is a scheme to efficiently encode, and therefore compress the network first. Fundamentally we find it a relevant issue to ask how much information is necessary to present a given network, and how symmetry can be exploited to this end.

In this chapter we will demonstrate one way to reduce the information storage of a network by using the idea that habitually graphs have many nodes that share many common neighbors (SBb08). So instead of recording all the links we could rather just store some of them and the difference between neighbors. The ideal compression ratio using this scheme will be $\eta = \frac{2}{<k>}$ where $< k >$ is the average degree of the network, compared to the standard compression using Yale Sparse Matrix Format (EGSS82) which gives $\eta_Y = \frac{1}{2} + \frac{1}{<k>}$. In practice this ratio is not attainable but the real compression ratio is still better than using YSMF as shown by our results.

8.1.2 Yale Sparse Matrix Format

Before going into our approach and data representation for sparse graphs, we first review a fundamental data representation for general sparse matrices, called the *Yale Sparse Matrix Format (YSMF)* (EGSS82). We illustrate first the standard form of YSMF by a simple example. Consider the matrix

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix},$$

it can be encoded using YSMF by a collection of 3 arrays:

$$\begin{aligned} VA &= [1,1,1,1,1,1,1,1], \\ IA &= [1,2,5,7,9], \\ JA &= [2,1,3,4,2,4,2,3], \end{aligned}$$

where VA contains the values of all non-zero entries of A, listed in the order from left to right then top to bottom of the matrix A; IA encodes the number of non-zero

entries of each row of A, $IA(1) := ||A(1,:)||_1$ and $IA(i+1) := IA(i) + ||A(i,:)||_1$ (for $i = 1, ..., N$); and JA contains the column indices of non-zero entries in A, listed in the same order as entries in VA.

However, for simple graphs, the corresponding adjacency matrices are symmetric and binary, thus it is sufficient to store only the upper (or lower) half of the matrix. Without loss of generality, we consider the upper half of A, so that $A(i,j) = 0$ whenever $i > j$. The array VA will consist of all 1's and thus is not necessary; also in IA we can let $IA(i) = ||A(i,:)||_1$ instead of the above definition so that IA contains exactly N entries; while JA is the same as the standard form of YSMF, which now contains M entries where M is the number of edges of the graph.

Thus, for an adjacency matrix of a simple graph, it requires $N + M$ entries to be stored using YSMF, which requires $\frac{N+M}{2}$ information units by the definition in the early sections in this paper. Compare to the original information storage M, we obtain the compression ratio $\frac{M+N}{2M}$.

8.2 Adjacency Matrix and Edge List

A graph $G = (V, E)$ is a set of vertices (or nodes) $V = \{v_1, v_2, ..., v_N\}$ together with edges (or links) $E = \{(v_i, v_j)\}$ which are the connected pairs. Graphs are often used to model networks. It is sometimes convenient to call the vertices that connect to a vertex i in a graph to be the neighbors of i. We will only consider undirected and unweighted graph in this chapter.

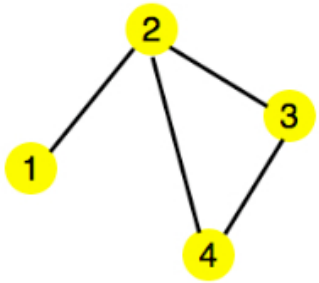

Figure 8.1. **A drawing of a planar embedding of an example graph.** -

A drawing as in Figure 8.1 allows us to directly visualize the graph (i.e. the nodes and the connections between them), but a truism that anyone who works with real world graphs from real data knows is that commonly those graphs are so large that even a drawing will not give any insight. Visualizing structure in graphs of such sizes ($N > 100$ to 1000) begs for some computer assistance.

An *adjacency matrix* is a common, although inefficient data representation of a graph. The adjacency matrix A_G of a graph $G = (V, E)$ is a $N \times N$ square matrix where N is the number of vertices of the graph and the entries $a_{i,j}$ of A_G are defined by:

$$a_{ij} = \begin{cases} 1, \text{ if nodes } i \text{ and } j \text{ are connected;} \\ 0, \text{ otherwise.} \end{cases} \tag{8.1}$$

For example, the adjacency matrix A_G for the graph in Figure 8.1 is

$$A_G = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}. \tag{8.2}$$

However, in the case that the number of edges in a graph are so few that the corresponding adjacency matrix is sparse, the *edge list* (or *adjacency list*) will be used instead. The edge list is a list of all the pairs of nodes that form edges in a graph. It is essentially the same as the edge set E for a graph $G = (V, E)$. Using edge list E_G to represent the same graph as above we will have:

$$E_G = \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{2, 4\}\}. \tag{8.3}$$

Note here that in the edge list we actually record the label of nodes for each edge in the graph, so for undirected graph, we can exchange the order for each pair of nodes.

We will only consider sparse simple graphs, whose adjacency matrices will thus be binary sparse matrices, and the standard information storage for such graphs or matrices will be the information units that are needed for the corresponding edge list (or two dimensional arrays).

We now sharpen the definition for the *unit of information* in our context. From the perspective of information theory (CT06), a message which contains N different symbols will require $log_2 N$ bits for each symbol, without any further coding scheme. The edge list representation is one example of a text file which contains N different symbols (often represented by natural numbers from 1 to N) for a graph containing N vertices. Note that the unit of information depends only on the number of symbols that appear in the message, i.e. the number of vertices in a graph, so for any given graph this will be a fixed number. Thus, when we restrict the disscussion to any particular graph, it is convenient to assume that each pair of labels in the edge list requires one information unit without making explicit what is the size of that unit. For example, the above graph (shown in Fig. 8.1) requires 4 information units. In this chapter we will focus on how to represent the same graph using fewer information units than its original representation.

8.3 A Motivating Example and the Idea of Redundancy

As a motivating example, let us consider the following graph (Fig. **??**)and its edge list.

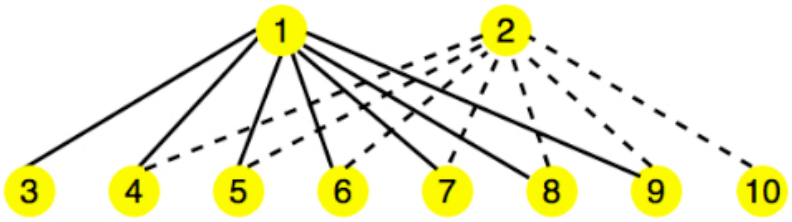

Figure 8.2. **An extreme example which shows similarity between vertices.** -

Note that here the neighbors of node 1 are almost the same as those of node 2. The edge list E_G for this graph will be:

$$E = \{\{1,3\},\{1,4\},\{1,5\},\{1,6\},\{1,7\},\{1,8\},\{1,9\},$$
$$\{2,4\},\{2,5\},\{2,6\},\{2,7\},\{2,8\},\{2,9\},\{2,10\}\}. \quad (8.4)$$

This requires 14 information units for the edge list. However, if we look back to the graph, we note that in this graph there are many common neighbors between node 1 and node 2, so there is a great deal of information redundancy. Considering the subgraphs, the neighbors of node 1 are almost the same as the neighbors of node 2, except that node 3 links to 1, but not 2, while node 10 links to 2, but not 1.

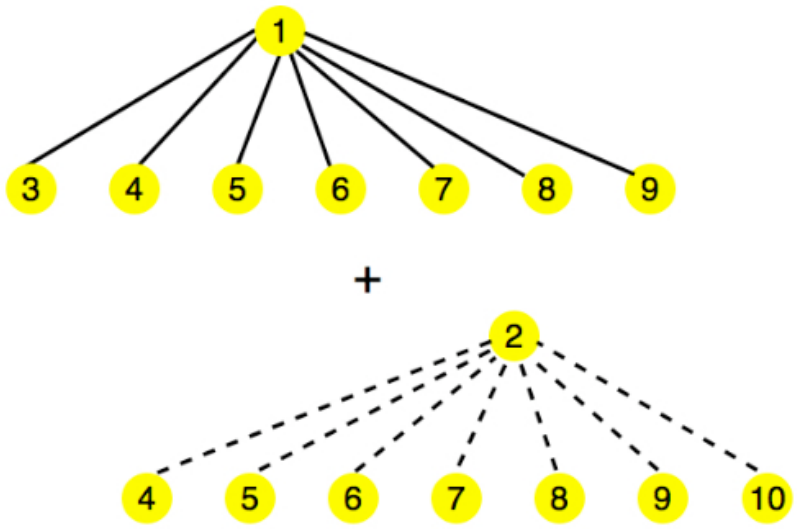

Figure 8.3. **Similar subgraphs of the original graph.** - Here the subgraph containing node 1 (on the top) is very similar to the one dominated by node 2 (on the bottom).

Taking the redundancy into account, we generate a new way to describe the same graph, exploiting the graphs. In the graph of Fig. 8.2, we see that the subgraph including vertices $1, 3, 4, 5, 6, 7, 8, 9$ is very similar to the subgraph including vertices $2, 4, 5, 6, 7, 8, 9, 10$, see Fig. 8.3. We exploit this redundancy in our coding.

We store the subgraph which only consists of node 1, and all its neighbors. Then, we add just two more parameters,

$$\alpha = (1, 2) \tag{8.5}$$

and

$$\beta = \{-3, 10\} \tag{8.6}$$

that allows us to reconstruct the original graph. Here the ordered pair $\alpha = (1, 2)$ tells us that in order to reconstruct the original graph we need to first copy node 1 to node

2. By copy, we mean the addition of a new node into the exsiting graph with label 2, and then linking all the neighbors of node 1 to the new node 2. See Fig. 8.4.

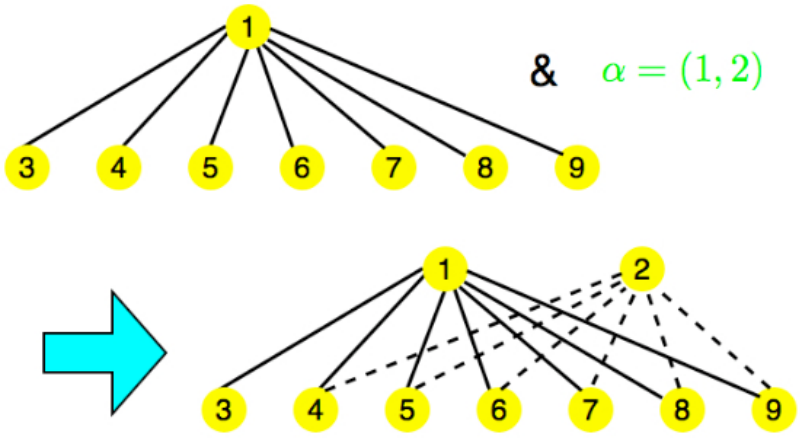

Figure 8.4. **Construct from the subgraph and parameter $\alpha = (1, 2)$. 'Copy' from node 1 to node 2.** -

The set $\beta = \{-3, 10\}$ tells us that we should then delete the link that connects the new node 2 and 3 and add a new link between 2 and 10. See Fig. 8.5.

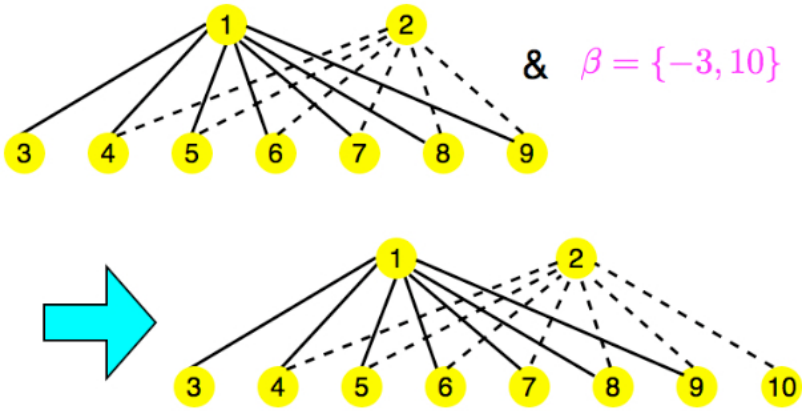

Figure 8.5. **Add and delete links according to $\beta = \{-3, 10\}$.** -

After all these operations we see that we successfully reconstruct the graph with fewer information units, in this case, nearly half as many as the original edge list. So instead of equation (8.4), we may use the edge list of the subgraph

$$E_{SG} = \{\{1, 3\}, \{1, 4\}, \{1, 5\}, \{1, 6\}, \{1, 7\}, \{1, 8\}, \{1, 9\}\} \tag{8.7}$$

as well as two sets

$$\alpha = (1, 2)$$
$$\beta = \{-3, 10\} \tag{8.8}$$

to represent the same graph.

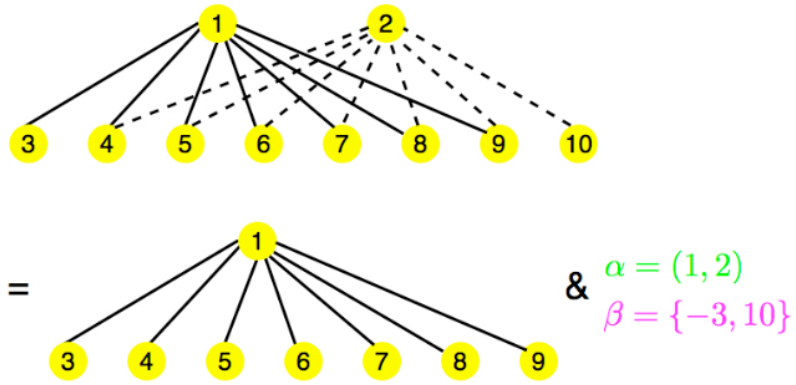

Figure 8.6. **Reconstruction of the original graph using a subgraph and the parameters α and β.** -

The above example suggests that by exploiting symmetry of the graph, we might be able to reduce the information storage for certain graphs by using a small subgraph as well as α and β as defined above.

However, there remains the question of how to choose the pair of vertices so that we actually reduce the information, and which is the best possible pair? It is important to answer these questions since most of the graphs are so large that we never will be able to see the symmetry just by inspection as we did for the above toy example.

In the following we answer the first question, and partly the second, by using a greedy algorithm. In section 4 we will define information redundancy for a binary sparse matrix and show that it reveals the neighbor similarity between vertices in a graph which is represented by its corresponding adjacency matrix. Then in section 5 we will give a detailed description of our algorithm which allows us to implement our main idea. Then in section 6 we will show some examples of these applications followed by discussion in section 7.

8.4 Information Redundancy and Compression of Sparse Matrices

Throughout this paper, we describe our methods in terms of manipulations of adjacency matrices to describe corresponding manipulations to the graphs. We choose this approach for pedagogical reasons, particularly regarding presentation of the analysis of information redundancy and algorithmic complexity. However, we emphasize that all of the necessary manipulations can and should be done in practice in terms of the more efficient edge list representation.

8.4.1 How to Choose Pairs of Vertices to Reduce Information

The graphs we seek to compress are typically represented by large sparse adjacency matrices. An edge-list is a specific data structure for representing such matrices, to reduce information storage. We will consider the edge-list form to be the standard way of storing sparse matrices, which requires M units of information for a graph with M edges. There are approaches of compressing sparse matrices, among which the most general is the Yale Sparse Matrix Format (YSMF, discussed in section 8.1.2), which does not make any assumption on the structure of the matrix and only requires $\frac{1}{2}(M+N)$ units of information. There are other approaches, such as (TY71) which emphasize not only the storage but also the cost for data access time. We will focus on the data storage, so the YSMF will be considered as a basic benchmark approach for compression of a sparse matrix, to which we will compare our results. The YSMF yields the compression ratio (see Appendix):

$$\eta_Y = \frac{M+N}{2M} = \frac{1}{2} + \frac{1}{< k >} \tag{8.9}$$

where $< k >= \frac{2M}{N}$ is the average degree of the graph.

We will show our approach of compressing the sparse matrices by first illustrating how the redundancy of a binary sparse matrix will be defined regarding to our specific operation on the matrix.

Generally, the adjacency matrix is a binary sparse matrix, $A = [a_{ij}]$ where a_{ij} equals 0 or 1 indicating the connectivity between node i and j. For a simple graph

consisting of M edges this matrix has $2M$ nonzero entries, but since it is symmetric only half of them are necessary to represent the graph, which yields M units of information for the edge-list.

Now, if two nodes i and j in the graph share a lot of similar neighbors, in the adjacency matrix row i and row j will have a lot of common column entries, and likewise for column i and column j (due to the symmetry of the matrix).

Suppose that we apply the operation to the graph, mentioned in the last section, by choosing $\alpha = (i, j)$ and the corresponding β, we will not need row j and column j in the matrix, to represent the graph. The number of nonzero entries in row j and column j is $2k_j$ where k_j is the degree of node j in the graph. By doing that, the number of nonzero entries in the new adjacency matrix becomes $2M - 2k_j$, which requires $M - k_j$ units of information. However, the extra information we have to record is encoded in α and β. α always has two entries, which requires 1 unit of information, and the units of information for β depend on the number of different neighbors between node i and node j. If i and j have Δ_{ij} different neighbors, the size of β will be

$$|\beta| = \Delta_{ij}, \tag{8.10}$$

and the units of information for β will thus be $\frac{1}{2}\Delta_{ij}$. Taking both the reduction of the matrix and the extra information into account, the actual information it requires after the operation is

$$M - k_j + 1 + \frac{1}{2}\Delta_{ij} = M - (k_j - 1 - \frac{1}{2}\Delta_{ij}). \tag{8.11}$$

This is true for i different from j. We could extend the operation to allow

$$\alpha = (i, i), \tag{8.12}$$

meaning a self-match, then we will put all the neighbors of i into the corresponding set β, and then delete these links associated with i. Then by a similar argument we find that after this operation we need

$$M - k_i + 1 + \frac{1}{2}k_i = M - (\frac{1}{2}k_i - 1) \tag{8.13}$$

units of information using the new format.

Note that here we need to clarify exactly the meaning of different neighbors since in the case that i and j are connected i is a neighbor of j but j is not, and likewise for j. However, this extra information can be simply encoded in α by making the following rule: $\alpha = (i, j)$ means when we reconstruct we do not connect i and j and $\alpha = (i, -j)$ means we connect i and j when we reconstruct. Then we can write $\Delta_{ij} = \|A(i,:) - A(j,:)\|_1 - 2a_{ij}$.

From the above discussion we see that if we define

$$r_{ij} = \begin{cases} k_j - 1 - \frac{1}{2}\Delta_{ij}, & \text{if } i \neq j; \\ \frac{1}{2}k_i - 1, & \text{if } i = j. \end{cases} \tag{8.14}$$

then by choosing $\alpha = (i, j)$, r_{ij} measures exactly the amount of information it reduces. We call r_{ij} the *information redundancy* between nodes i and j. Note here that in general this redundancy is not symmetric in i and j, since for any pair of nodes Δ_{ij} is symmetric but the degree of these two nodes can be different, and deleting the node with higher degree will always reduce more units of information compared to deleting the lower degree node.

We form the redundancy matrix R by setting the entry in row i and columnn j to be r_{ij}. We perform the shrinking operation for the pair with maximum r_{ij}, thus saving the maximum amount of information.

For example, again using the graph from section 2, the adjacency matrix is:

$$A = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{8.15}$$

and the corresponding redundancy matrix is:

$$R = \begin{bmatrix} 2.5 & 5 & -3 & -2.5 & -2.5 & -2.5 & -2.5 & -2.5 & -2.5 & -4 \\ 5 & 2.5 & -4 & -2.5 & -2.5 & -2.5 & -2.5 & -2.5 & -2.5 & -3 \\ 3 & 2 & -0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & -1 \\ 2.5 & 2.5 & -0.5 & 0 & 1 & 1 & 1 & 1 & 1 & -0.5 \\ 2.5 & 2.5 & -0.5 & 1 & 0 & 1 & 1 & 1 & 1 & -0.5 \\ 2.5 & 2.5 & -0.5 & 1 & 1 & 0 & 1 & 1 & 1 & -0.5 \\ 2.5 & 2.5 & -0.5 & 1 & 1 & 1 & 0 & 1 & 1 & -0.5 \\ 2.5 & 2.5 & -0.5 & 1 & 1 & 1 & 1 & 0 & 1 & -0.5 \\ 2.5 & 2.5 & -0.5 & 1 & 1 & 1 & 1 & 1 & 0 & -0.5 \\ 2 & 3 & -1 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & -0.5 \end{bmatrix}, \quad (8.16)$$

The maximum entry in R is $r_{12} = r_{21} = 5$, indicating that either choice of $\alpha = (1, 2)$ or $\alpha = (2, 1)$ will give the maximum information reduction, and the corresponding β can be obtained by recording the column entries in row 1 and row 2 according to our rule.

In the above discussion we only consider a one step shrinking operation on the graph and find out the direct relationship between the maximum information reduction and the redanduncy matrix. But we know that after deleting one node the resulting graph is still sparse and so could be compressed further by our scheme. The question is then how to successively choose α and β to obtain the best overall compression.

8.4.2 On Greedy Optimization of The α, β, Orbit

Let $\alpha_t = (i_t, j_t)$ denote the operation at step t, $t = 1, 2, ..., T$ (here the sign for j_t would not affect our analysis so by convience we just write j_t). In order to analyze the multi-step effect, we first consider how the adjacency matrix A is affected by the orbit $\{\alpha_t\}$. Let $A_0 = A$ be the original adjacency matrix. Let A_t be the corresponding adjacency matrix after applying α_t and the entries in it be $A_t(i, j)$. On deleting node j_t we actually set row and column j_t to be zero in A_{t-1} and all the other entries are

unchanged, to obtain the new matrix A_t, i.e.

$$A_t(i,j) = \begin{cases} A_{t-1}(i,j), & \text{if } i \neq j_t \text{ and } j \neq j_t; \\ 0, & \text{if } i = j_t \text{ or } j = j_t. \end{cases} \tag{8.17}$$

So by induction we see that

$$A_t(i,j) = \begin{cases} A_0(i,j), & \text{if } i \notin \{j_1, ..., j_t\} \text{ and } j \notin \{j_1, ..., j_t\}; \\ 0, & \text{if } i \in \{j_1, ..., j_t\} \text{ or } j \in \{j_1, ..., j_t\}. \end{cases} \tag{8.18}$$

Then we analyze how the *redundancy matrix* R changes. Use R_t to represent the redundancy matrix, $k_t(i)$ the degree of node i, and $\Delta_t(i,j)$ the number of different neighbors of node i and j, associated with the graph of A_t. Since our goal is to achieve compression, once a node is deleted in the graph it is useless for future operations. So we will set $R_t(i,j) = 0$ if i or j has been deleted before, i.e.

$$R_t(i,j) = 0 \qquad \text{if } i \in \{j_1, ..., j_t\} \text{ or } j \in \{j_1, ..., j_t\}. \tag{8.19}$$

Now for those i and j that have not been deleted, i.e. $i, j \notin \{j_1, ..., j_t\}$, by equation 8.14 we see that $R_t(i,j) = k_t(j) - 1 - \frac{1}{2}\Delta_t(i,j)$ for $i \neq j$ and $R_t(i,i) = \frac{1}{2}k_t(i) - 1$. Since A_t is obtained by deleting row and columnn j_t in A_{t-1}, the degree of each node changes according to:

$$k_t(i) \;=\; k_{t-1}(i) - A_{t-1}(i, j_t) \tag{8.20}$$

and Δ_{ij} changes according to

$$\Delta_t(i,j) \;=\; \Delta_{t-1}(i,j) - |A_{t-1}(i,j_t) - A_{t-1}(j,j_t)| \tag{8.21}$$

Thus, we conclude that for $i \neq j$

$$\begin{aligned} R_t(i,j) &= k_{t-1}(j) - A_{t-1}(j,j_t) - 1 - \frac{1}{2}[\Delta_{t-1}(i,j) - |A_{t-1}(i,j_t) - A_{t-1}(j,j_t)|] \\ &= k_{t-1}(j) - 1 - \frac{1}{2}\Delta_{t-1}(i,j) - A_{t-1}(j,j_t) + \frac{1}{2}|A_{t-1}(i,j_t) - A_{t-1}(j,j_t)| \\ &= R_{t-1}(i,j) + [\frac{1}{2}|A_{t-1}(i,j_t) - A_{t-1}(j,j_t)| - A_{t-1}(j,j_t)] \end{aligned} \tag{8.22}$$

and for $i = j$

$$\begin{aligned} R_t(i,i) &= \frac{1}{2}k_t(i) - 1 \\ &= \frac{1}{2}(k_{t-1}(i) - A_{t-1}(i,j_t)) - 1 \\ &= R_{t-1}(i,i) - \frac{1}{2}A_{t-1}(i,j_t). \end{aligned} \tag{8.23}$$

By induction, we obtain that for $i \neq j$:

$$R_t(i,j) = R_0(i,j) + \sum_{\tau=1}^{t}[\frac{1}{2}|A_{\tau-1}(i,j_\tau) - A_{\tau-1}(j,j_\tau)| - A_{\tau-1}(j,j_\tau)] \tag{8.24}$$

and for $i = j$:

$$R_t(i,i) = R_0(i,i) + \sum_{\tau=1}^{t}[-\frac{1}{2}A_{\tau-1}(i,j_\tau)]. \tag{8.25}$$

By use oft the fact that $i, j \notin \{j_1, ..., j_t\}$, by equation 8.17, we can simplify the above two expressions to yield,

$$R_t(i,j) = \begin{cases} R_0(i,j) + \sum_{\tau=1}^{t}[\frac{1}{2}|A_0(i,j_\tau) - A_0(j,j_\tau)| - A_0(j,j_\tau)], & \text{if } i \neq j; \\ R_0(i,i) + \sum_{\tau=1}^{t}[-\frac{1}{2}A_0(i,j_\tau)], & \text{if } i = j. \end{cases} \tag{8.26}$$

Note that if we choose a pair (i_t, j_t) at step t, the information we save is measured by $R_{t-1}(i_t, j_t)$. Thus, for any orbit $\{\alpha_t = (i_t, j_t)\}_{t=1}^{T}$ satisfying $i_t, j_t \notin \{j_1, ..., j_{t-1}\}$ for $t = 2, 3, ..., T$ (we call such an orbit a *natural orbit*), the total information reduction (or information saving) will be:

$$\begin{aligned} s(\{\alpha_t\}_{t=1}^{T}) &= \sum_{t=1}^{T} R_{t-1}(i_t, j_t) \\ &= \sum_{t=1}^{T}[R_0(i_t, j_t) + c(i_t, j_t, t)] \end{aligned} \tag{8.27}$$

where c is defined by:

$$c(i,j,t) = \begin{cases} \sum_{\tau=1}^{t}[\frac{1}{2}|A_0(i,j_\tau) - A_0(j,j_\tau)| - A_0(j,j_\tau)], & \text{if } i \neq j; \\ \sum_{\tau=1}^{t}[-\frac{1}{2}A_0(i,j_\tau)], & \text{if } i = j. \end{cases} \tag{8.28}$$

So the compression problem can be stated as:

$$\text{Find} \max_{\{\alpha_t\}_{t=1}^{T}} s(\{\alpha_t\}_{t=1}^{T}). \tag{8.29}$$

One more thing to mention is that the length of the orbit, T, is also a variable, which could not be larger than N since there are only N nodes in the graph and it is meaningless to delete an 'empty' node which does not even exist.

8.5 Greedy Algorithm for Compression

From the previous section we see that for a given adjacency matrix, the final compression ratio depends on the orbit $\{\alpha_t\}_{t=1}^{T}$ we choose, and the compression problem becomes an optimization problem. However, to find the maximum of s and the corresponding best orbit is not trivial. One reason is that the number of natural orbits is of order $N!$, which makes it impractical to test and try for all possible orbits. Another reason which is crucial here is that for any given orbit of length T, evaluating s costs $O(T^2)$ operations, making it hard to find an appropriate scheme to search for the true maximum or even the approximate maximum. Instead, we use a greedy algorithm to find an orbit which gives a reasonable compression ratio, and which is easy to apply.

The idea of the greedy algorithm is that at each iteration step we choose the pair of nodes i_t and j_t which maximizes $R_{t-1}(i,j)$ over all possible pairs, and we stop if the maximum value is non-positive. Also we need to record α and β according to the graph.

Here we summarize the greedy algorithm as pseudocode. Given the adjacency matrix A of a graph (N nodes and M edges).

1. Initialization:
   (a) Set $A_0 = A$.
   (b) Calculate $R_0(i,j)$ for all $i,j = 1,...,N$. This forms the redundancy matrix $R_0 = R$.
   (c) Set $t = 1$.

2. Greedy compression:

   (a) Let $R_{t-1}(i_t, j_t)$ be the largest element in R_{t-1}.
   If $R_{t-1}(i_t, j_t) > 0$
   record $\alpha_t = (i_t, j_t)$,
   then go to step $2(b)$.
   Else,
   Terminate.

   (b) Set β_t according to the difference between the two rows of α_t in A_{t-1},
   Update A_{t-1} to A_t according to (8.17);
   Update R_{t-1} to R_t according to (8.22) and (8.23) for $i, j \neq j_t$;
   Set $R_t(i, j) = 0$ for i or $j = j_t$.

   (c) Set $t = t + 1$ and go to step $2(a)$.

The compressed version of the matrix will consist of: the final matrix A_T, the orbit $(\alpha_1, ..., \alpha_T)$ and the vectors $\{\beta_1, ..., \beta_T\}$, which will allow us to reconstruct $A = A_0$ and any intermediate matrix A_t during the compression process.

The computational complexity of this greedy algorithm is dominated by the initialization of the redundancy matrix R, which requires $O(MN)$ operations [1]. The subsequent operations will each require only an update of R according to formula (8.22) and (8.23), result in $O(N)$ operations per step. Thus the overall cost of the greedy algorithm will be $O(MN)$ and the average cost per step is $O(\frac{MN}{T})$ where T is the number of shrinking steps ($T < N$).

8.6 Examples of Application to Graphs

In this section we will show some examples of our compression scheme on several networks. We begin with the lattice graph, which is expected to be readily compressible due to the high degree of overlapping between neighbors of nodes. As a secondary example, we add some random alterations, and apply our method to the

[1] Here $O(MN)$ comes from the fact that we are comparing N^2 pair of vertices, and each comparison requires $O(\frac{M}{N})$ operations (in efficient format like the edge-list) since the matrix is sparse. Thus the cost for initializing R is $O(N^2 \cdot \frac{M}{N}) = O(MN)$

corresponding Watts-Strogatz network. Finally we show some results for real-world networks.

8.6.1 A Simple Benchmark Example: Lattice Graph

One of the most symmetric graphs is the lattice graph, a one-dimensional chain where each site is connected to $\frac{k}{2}$ nearest neighbors to its right and left. In this case $< k >= k$ represents the degree of each vertex in the lattice graph. The total number of nodes is $N >>< k >$, the corresponding adjacency matrix is sparse.

We implement our algorithm for lattice graph with different $< k >$. The results are shown in Fig. 8.7. Here we take $N = 500$.

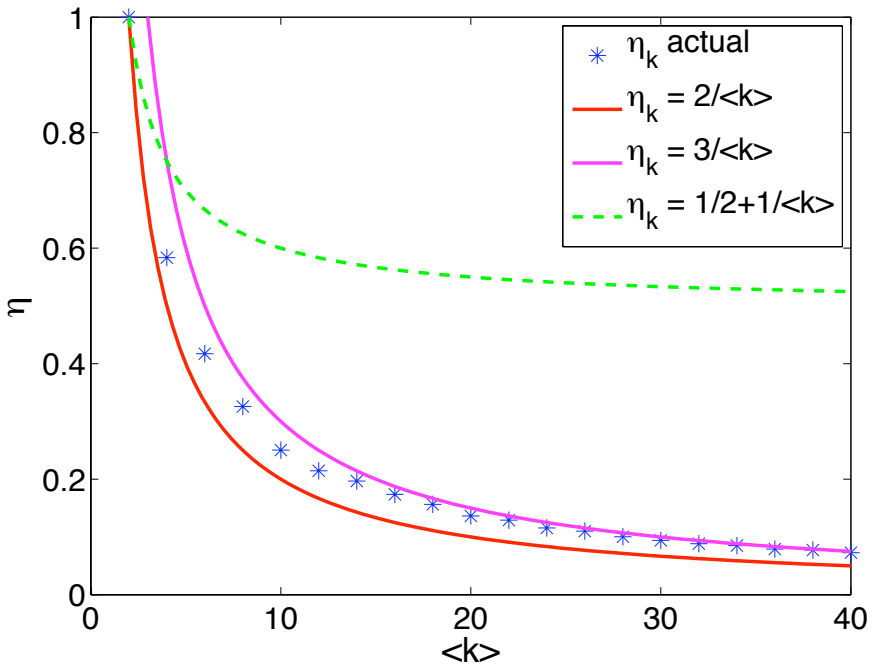

Figure 8.7. **Compression results for lattice graphs.** - Stars indicate the final compression ratios for the lattice graphs with $< k > 2$ to 40. The compression limit is indicated by the bottom curve given by $\eta_k = \frac{2}{<k>}$, and we find that for $< k >$ large the compression ratio is close to the empirical formula: $\eta_k = \frac{3}{<k>}$ (upper curve). For comparison, we plot the result using YSMF (broken line): $\eta_k = \frac{1}{2} + \frac{1}{<k>}$. For $< k >> 2$, our algorithm always achieves a better result than the YSMF and the advantage increases with increasing $< k >$.

8.6.2 Compressing a Watts-Strogatz Small-World Graph

It is not surprising that the lattice graphs are easy to compress since these graphs are highly symmetric and nodes have lots of overlaps in their neighbors. However, in the case that we don't have such perfect symmetry, we still hope to achieve compression. Here we apply our algorithm to the WS graphs. The WS graph comes from the famous Watts-Strogatz model for real-world networks by showing the so called small-world phenomenon. The WS graph is generated from a lattice graph by the usual rewiring of each edge with some given probability p from the uniform distribution.

We apply our algorithm to WS graphs with different p to explore how p affects the compression behavior. Results are shown in Fig. 8.8.

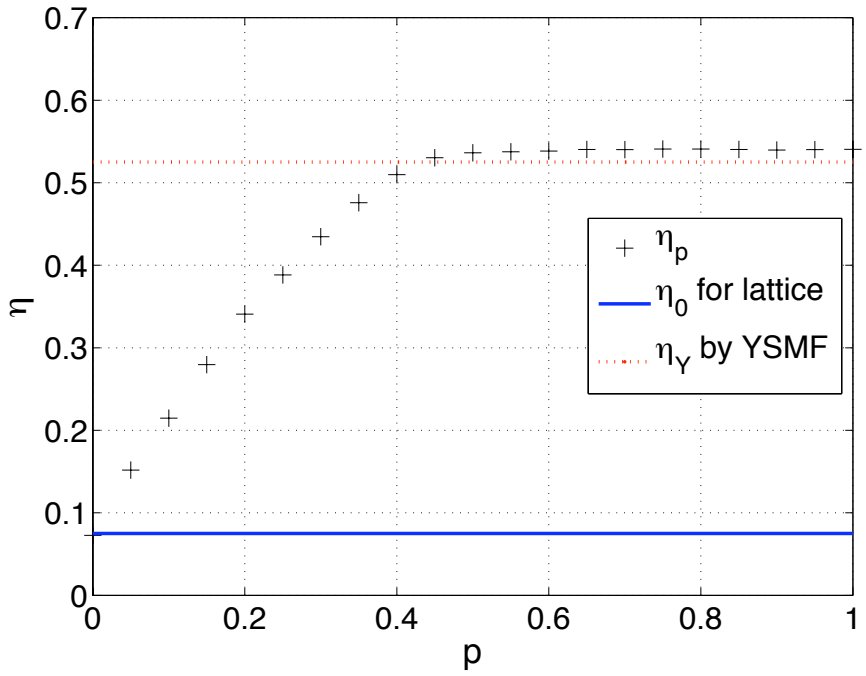

Figure 8.8. **Compression results for WS graphs.** - Here the base lattice graph is with $N = 500$ and $< k >= 40$. The stars show the compression results by our algorithm. The lower line is the compression ratio for the lattice $N = 500$ and $< k >= 40$ and the upper line is the ratio from the YSMF. We see that as p increases there is less and less overlapping between neighbors in the network and the compression ratio increases. For $p \sim 0.5$, we obtain worse result than YSMF.

8.6.3 Real-World Graphs

In the following we show the compression results for some real world graphs: a C.elegans metabolic network (DA05) (Fig. 8.9), a yeast network constructed from yeast reactions (BZC$^+$03), an email network (GDD$^+$03), and an airline network of flight connections (Paj). In the Table 1 we summarize the compression results for these real world graphs.

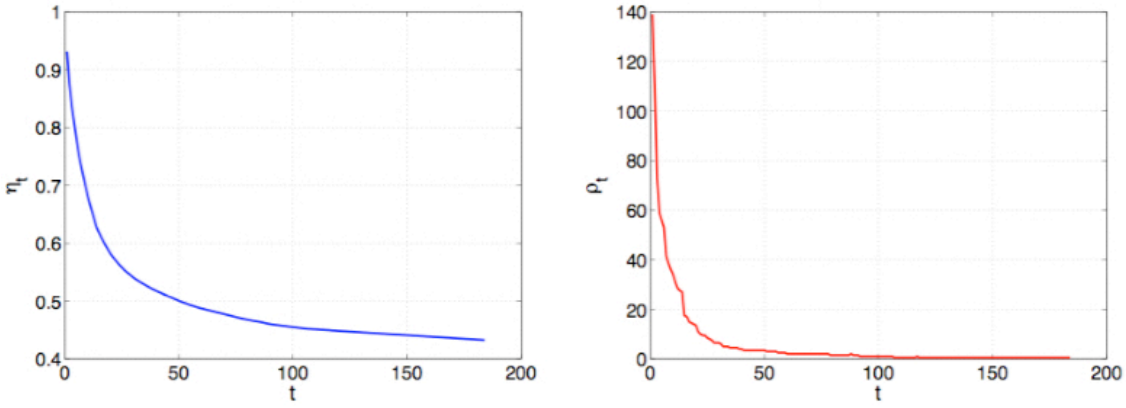

Figure 8.9. **Compression process for Metabolic network (DA05).** - Compression ratio η during each step (left), and information redundancy ρ each step (right).

Network	N	$< k >$	η_Y	η	η_*
Lattice	N	$< k >$	$\frac{1}{2} + \frac{1}{<k>}$	$\frac{3}{<k>}$	$\frac{2}{<k>}$
Yeast (BZC$^+$03)	2361	6.08	0.66	0.50	0.33
Metabolic (DA05)	453	9.01	0.61	0.43	0.22
Email (GDD$^+$03)	1133	9.62	0.60	0.49	0.21
Airline (Paj)	332	12.81	0.58	0.31	0.16

Table 8.1. Compression results for some networks.

8.7 Discussion and Open Problems

From the previous section we see that our algorithm works for various kinds of graphs and gives a reasonable result. The ideal limit of our method for a graph with N nodes, M edges and average degree $< k >= \frac{2M}{N}$, which is relative large, is $\frac{2}{<k>}$. This is obtained when each β_t during the compression process is empty, meaning that

most of the nodes share common neighbors, in which case we only need to record all the α_t, requiring $\frac{N}{2}$ units of information and yields

$$\eta = \frac{N}{2M} = \frac{2}{< k >}. \tag{8.30}$$

Notice that trees do not compress, since for trees $< k >= 2$, so on average the overlap in neighbors will be even smaller (likely to be 0), and a possible way to achieve compression is by self-matching for large degree nodes, for example, the hubs in a star graph. For comparison, the YSMF always gives the compression ratio

$$\eta_Y = \frac{1}{2} + \frac{1}{< k >} \tag{8.31}$$

which does not compress trees, and has a lower bound $\frac{1}{2}$, while our method in principle approaches 0 as $< k > \to \infty$. Actually the compression ratio using YSMF can be achieved by choosing a special orbit in our approach which only contains self-matches α, i.e.

$$\{\alpha_t\}_{t=1}^{T} = \{(i,i)\}_{i=1}^{N}. \tag{8.32}$$

In this case the neighbors of each node will be put into corresponding β sets and since any α_i contains the same pair of numbers (i,i) we can just use one i to represent the pair, resulting in a total $\frac{N+M}{2}$ information units. So our approach can be considered as a generalization of the YSMF.

However, as we observed in our compression results, the compression ratio given by 8.30 is in general not attainable since it is only achieved for the ideal case that nearly every node in the graph shares the same neighbors, and yet the graph needs to be sparse! However, for lattices we observe that the actual compression ratio achieved by our algorithm is about $\frac{3}{<k>}$, which is of the same order as the ideal compression ratio. For WS graphs, when the noise p is small, our algorithm achieves better compression ratio than YSMF, and the compression ratio is nearly linearly dependent on p for $p < 0.5$. For $p > 0.5$ the graph resembles Erdos-Renyi random graphs (Bol01), there is no symmetry between nodes to be used and thus our approach does not give good result, as compared to the YSMF.

For real world graphs, the results by our algorithm are better than using YSMF, but not as good as we observed for lattice graphs. This suggests that in real world graphs nodes, in general, share certain amount of common neighbors even when the

total number of links is small. This kind of overlap in neighbors is certainly not as common as we see in lattice graphs since real world graphs in general have more complicated structures. The problem of compressing a general sparse graph has close connection to the problem of defining appropriate entropy measure for certain class of random graph models. The paper (CS09) addresses such problem for Erdos-Renyi graphs. Finding a general approach for graphs that seem to be more similar as real world networks is still an open problem.

Chapter 9

Sequence Networks

9.1 Background

9.1.1 Threshold Graph

A *threshold graph* $G = (V, E)$ ($V = \{1, 2, ..., n\}$) is the type of graph where each node $i \in V$ in the graph is assigned some (hidden) weight x_i, and an edge exists between node i and j if and only if the sum of their weights $x_i + x_j$ exceeds a certain threshold θ, i.e.,

$$(i, j) \in E \Leftrightarrow x_i + x_j > \theta. \tag{9.1}$$

Fig. 9.1 shows, for the given weights $\{0.5, 0.4, 0.2, 0.2, 0.1\}$, different threshold graphs for the different choices of the threshold θ. Note that when the threshold increases, the graph changes from a complete graph where every one is connected, to an empty graph where no edge exists.

The original threshold graph model and its variants have been studied extensively in the area of graph theory and linear algebra (Gol80; HIS81; Mer94; Mer03), and later on used by statistical phycists to develop various models for complex networks (BP03; CCDM02; KMRS05; MMK04).

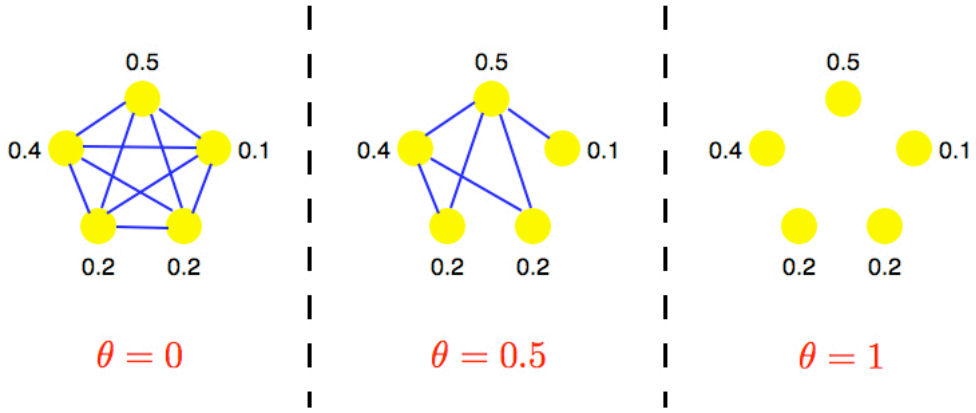

Figure 9.1. **Example of threshold graphs.** - Here the numbers are the weights x_i. Two nodes are connected, for a given θ, if and only if $x_i + x_j > \theta$. On the left, for $\theta = 0$, the corresponding threshold graph is a complete graph, whereas on the right (when $\theta = 1$) no edge appears; in the middle we have a nontrivial threshold graph, for $\theta = 0.5$.

9.1.2 Creation Sequence

In 2006, Hagberg *et al.* reported an interesting observation that a threshold graph can be represented compactly by a sequence of numbers (HSS06), called the *creation sequence*. The creation sequence is a binary sequence of numbers $S = (s_1, ..., s_n)$. For a given creation sequence of length n, a graph of n nodes is constructed, where the presence of specific edges are determined by the creation sequence: for nodes $i < j$,

$$(i, j) \in E \Leftrightarrow s_j = 1. \tag{9.2}$$

For example, the threshold graphs in Fig. 9.1 correspond to the creation sequences

$$S = \begin{cases} (0_{0.1}, 1_{0.2}, 1_{0.2}, 1_{0.4}, 1_{0.5}), & \text{for } \theta = 0; \\ (0_{0.2}, 0_{0.2}, 1_{0.4}, 0_{0.1}, 1_{0.5}), & \text{for } \theta = 0.5; \\ (0_{0.5}, 0_{0.4}, 0_{0.2}, 0_{0.2}, 0_{0.1}), & \text{for } \theta = 1. \end{cases} \tag{9.3}$$

Here the subscripts are used to specify the correspondence between numbers in the creation sequence and weights on nodes, in general such subscripts are unnecessary.

For a given threshold graph with weights $\{x_1, ..., x_n\}$ (without loss of generality, suppose that $x_1 \leq x_2 \leq ... \leq x_n$) and threshold θ, its unique creation sequence can be obtained by performing the following algorithm (HSS06):

- Let $i = 1, j = n$, and $k = n$.

  1. If $x_i + x_j \leq \theta$

     Set $s_k \to 0, k \to k - 1$ and set $i \to i + 1$.

     Otherwise (i.e., if $x_i + x_j > \theta$)

     Set $s_k \to 1, k \to k - 1$ and set $j \to j - 1$.

  2. Terminate when $k = 1$, and set $s_1 \to 0$; Otherwise, go to step 1.

[You may want to try it for the threshold graphs shown in Fig. 9.1 and check with the results stated in Eq. (9.3).]

Note that the first digit in the creation sequence is arbitrary (either choice of 0 or 1 yields the same graph). Other than the first digit, the creation sequence for a given threshold graph is *unique*. Also, by definition, for a given creation sequence one can construct a unique threshold graph that corresponds to it. Thus, by the above argument, a one-to-one correspondence exists between the space of threshold graphs and their creation sequences.

The creation sequence uniquely encodes a threshold graph, with only n storage units. Moreover, it was shown in (HSS06) that computation of important network statistics such as degree, clustering, betweenness, and Laplacian spectrum can be performed directly on the creation sequence, usually in linear time.

Such advantages can be further utilized for the design of large networks to satisfy certain statistical properties, such as the degree distribution and eigen-ratio (defined as the ratio of the second smallest eigenvalue versus the largest eigenvalue) of the graph Laplacian.

However, since the type of creation sequence is introduced specifically for threshold graphs, the limitation is obvious. For example, the diameter of a threshold graph is either 1 or 2, regardless of the number of nodes in the graph.

In this chapter we will introduce a broader class of networks that can be constructed from a sequence of letters, called sequence networks (SNb08). With this generalization, a wider range of graphs can be described by the sequence representation which allows high compressibility and fast computation of statistics and more flexibility to model more general graphs as opposed to the class of threshold graphs.

In section 9.2 we define sequence networks in a general sense and show how a threshold graph can be map into a special class of two-letter sequence networks. We fully classify all the two-letter and three-letter sequence networks and study some examples of sequence networks in sections 9.3 and 9.4. Discussion and open problems are included in section 9.5.

9.2 Generalization: Sequence Networks

In view of the construction of a graph given some creation sequence, such as the rule determined by (9.2), the key is to decide, for each letter (0 or 1 in the binary case), which nodes to connect to. For example, a creation sequence $S = \{s_1, ..., s_n\}$ can be used to construct a threshold graph by starting with an empty graph with only one node and set $k = 2$, and the following recursive construction rule:

1. If $s_k = 1$
   Add a node (indexed by k) to the existing graph;
   Connect node k with all other nodes $i = 1, 2, ..., k-1$.
   Otherwise (i.e., $s_k = 0$)
   Add a node (indexed by k) to the existing graph.

2. If $k < n$
   Set $k \to k + 1$;
   Otherwise
   Terminate.

Using this algorithm, one can easily verify that the creation sequences in Eq. (9.3) indeed generate graphs shown in Fig. 9.1.

The key in this construction is the rule specified by the *if* statement in step 1, which basically says that whenever a node k with $s_k = 1$ is added to the existing graph, it connects to all the preexisting nodes whereas when a node k with $s_k = 0$ is added, it connects to nothing (although it might be connected later on by newly added nodes).

In the following of this chapter, we will switch our notation for creation sequence from binary numbers 0 and 1, to letters A and B, to avoid confusion which might occur in the indexing of a node (usually by a natural number) and the indexing of the *type* of a node (specified by s_k where k is the index of that node). So for example, a creation sequence previously presented as $(0, 0, 1, 0, 1)$ will now (and later on) be denoted by (A, A, B, A, B). Similarly when there are more than two letters.

From a binary sequence of numbers $S = (s_1, ..., s_k)$, one can follow a similar construction but with different rules of connecting nodes $s_k = A$ and $s_k = B$. For example, one could specify that when node of type A is added, it connects to all the existing nodes of type A; and when a type B node is added; it connects to all the nodes of type B. This construction rule will in general result in a disconnected graph of two *cliques* (a clique is a complete subgraph), corresponding to nodes of type A and B respectively. Such a rule can be represented by the following *rule matrix* where the first and second indices correspond to A and B:

$$\mathbf{R}_3 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{9.4}$$

where the first row $\mathbf{R}_3(1,:) = (1\ 0)$ indicates that nodes of type A, when added, connects to existing A's, but not B's; while the second row $\mathbf{R}_3(2,:) = (0\ 1)$ indicates that nodes of type B, when added, do not connect to existing A's, but to B's. Therefore, the rule matrix used to generate threshold graphs from creation sequences can be specified by:

$$\mathbf{R}_4 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}. \tag{9.5}$$

In general, when a creation sequence is made up of two letters, there are 2^4 possible rule matrices, since a $2 - by - 2$ matrix has 4 entries, each can be either 0 or 1. When the number of letters increase, there is a lot more different rule matrices. For example, there are possibly $2^9 = 512$ rule matrices for three-letter sequence networks, and $2^16 = 65536$ ones for four-letter sequence networks, etc.

9.3 Classification of Two-Letter Sequence Networks

As discussed in the previous section, for graphs that can be constructed from sequences $S = (s_1, s_2, \ldots, s_n)$ of the two letters A and B, there are 16 possible rules. Fig. 9.2 gives an example of the graph obtained from the sequence (A, A, A, B, B, A, A, B), applying the *threshold* rule $\binom{0\,0}{1\,1}$.

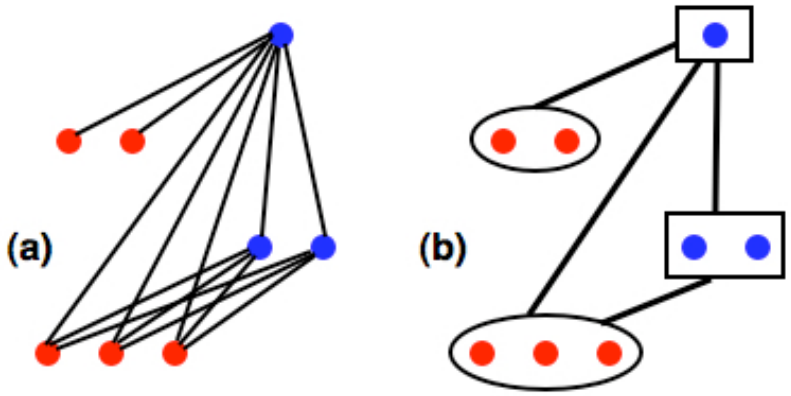

Figure 9.2. **Layer representation of threshold graph.** - Threshold graph: (a) The threshold graph resulting from the sequence (A, A, A, B, B, A, A, B), and (b) its box representation, highlighting modularity. Nodes are added one at a time from bottom to top, A's on the left and B's on the right.

In this section we fully classify all the two-letter sequence networks, and show that most of them can be reduced to one of the three basic fundamental two-letter sequence network types.

9.3.1 Classification

To start with, we shall disregard the four rules that fail to connect between A and B (and thus fail to generate connected graphs)

$$\mathbf{R}_0 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \ \mathbf{R}_1 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \ \mathbf{R}_2 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \ \mathbf{R}_3 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad (9.6)$$

for they yield simple *disjoint* graphs of the two types of nodes: $\mathbf{R}_0$ yields isolated nodes only, $\mathbf{R}_3$ yields one complete graph of type A and one of type B, $\mathbf{R}_1$ yields a complete graph of type A and isolated nodes of type B, etc.

The list of remaining rules can be shortened further by considering two kinds of symmetries: (a) permutation, and (b) time reversal. *Permutation* is the symmetry obtained by permuting between the two types of nodes, $A \leftrightarrow B$. Thus, a permuted rule ($R_{11} \leftrightarrow R_{22}$ and $R_{12} \leftrightarrow R_{21}$) acting on a permuted sequence $(\bar{s}_1, \bar{s}_2 \ldots, \bar{s}_n)$ yields back the original graph, where $\bar{s}_k$ stands for the inverted type: $\bar{s}_k = A$ if $s_k = B$, and vice versa. *Time reversal* is the symmetry obtained by reversing the arrows ("time") in the connectivity rules, or taking the transpose of $\mathbf{R}$. The transposed rule (or equivalently, the transpose rule matrix R^T) acting on the reversed sequence $(s_n, s_{n-1}, \ldots, s_1)$ yields back the original graph. The two symmetry operations are their own inverse and they form a symmetry group. In particular, one may combine the two symmetries: a rule with $R_{11} \leftrightarrow R_{22}$ applied on a reversed sequence with inverted types $(\bar{s}_n, \bar{s}_{n-1}, \ldots, \bar{s}_1)$ yields back the original graph, see Fig. 9.3.

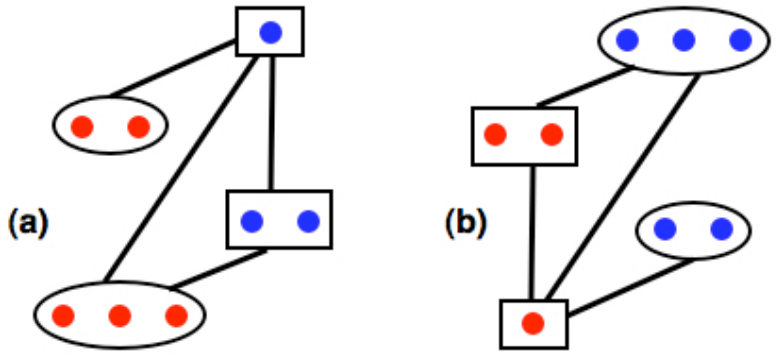

Figure 9.3. **Combined time reversal and permutation symmetry for sequence networks.** - The graphs resulting from $\mathbf{R}_4$ applied to the sequence (A, A, A, B, B, A, A, B) (a), and from $\mathbf{R}_6$ applied to the reverse-inverted sequence (A, B, B, A, A, B, B, B) (b), are identical.

All of the four rules

$$\mathbf{R}_4 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \ \mathbf{R}_5 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}, \ \mathbf{R}_6 = \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}, \ \mathbf{R}_7 = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}, \qquad (9.7)$$

are equivalent and generate threshold graphs. $\mathbf{R}_4$ is the rule for threshold graphs exploited by Hagberg *et al.*, (HSS06), and $\mathbf{R}_5$ is equivalent to it by permutation. $\mathbf{R}_6$

is obtained from $\mathbf{R}_4$ by time reversal and permutation (Fig. 9.3), and $\mathbf{R}_7$ is obtained from $\mathbf{R}_4$ by time reversal.

The two rules

$$\mathbf{R}_8 = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, \ \mathbf{R}_9 = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \tag{9.8}$$

are equivalent, by either permutation or time reversal, and generate non-trivial bipartite graphs that are different from threshold graphs (Fig. 9.4).

The rule $\mathbf{R}_{10} = \binom{0\,1}{1\,0}$ generates complete bipartite graphs. However, the complete bipartite graph $K_{p,q}$ can also be produced by applying $\mathbf{R}_8$ to the sequence $(A, A, \ldots A, B, B, \ldots B)$ of p A's followed by q B's, so the rule $\mathbf{R}_{10}$ is a "degenerate" form of $\mathbf{R}_8$. One could see that this is the case at the outset, because of the symmetrical relations $A \to B$, $B \to A$: these render the ordering of the A's and B's in the graph's sequence irrelevant. By the same principle, $\mathbf{R}_{11} = \binom{0\,1}{1\,1}$ and $\mathbf{R}_{12} = \binom{1\,1}{1\,0}$ are degenerate forms of $\mathbf{R}_4$ and $\mathbf{R}_5$, respectively. They yield threshold graphs with segregated sequences of A's and B's.

The two rules

$$\mathbf{R}_{13} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \ \mathbf{R}_{14} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \tag{9.9}$$

are equivalent, by either permutation or time reversal, and generate non-trivial graphs different from threshold graphs and graphs produced by $\mathbf{R}_8$ (Fig. 9.4). Finally, the rule $\mathbf{R}_{15} = \binom{1\,1}{1\,1}$ is a degenerate form of $\mathbf{R}_{13}$ (or $\mathbf{R}_{14}$) and yields only complete graphs (which are threshold graphs, so $\mathbf{R}_{15}$ is subsumed also in $\mathbf{R}_4$).

To summarize, $\mathbf{R}_4$, $\mathbf{R}_8$, and $\mathbf{R}_{13}$ are the only two-letter rules that generate different classes of non-trivial connected graphs. There is yet another amusing type of symmetry: applying $\mathbf{R}_8$ and $\mathbf{R}_{13}$ to the same sequence yields *complement*, or *inverse* graphs — nodes are adjacent in the inverse graph if and only if they are *not* connected in the original graph. The figure-background symmetry manifest in the rules $\mathbf{R}_8$ and $\mathbf{R}_{13}$ ($0 \leftrightarrow 1$) is also manifest in the graphs they produce (Fig. 9.4a,c). On the other hand, the inverse of threshold graphs are also threshold graphs. Also, the complement of a threshold rule applied to the complement (inverted) sequence yields back the original graph. In this sense, threshold graphs have maximal symmetry. $\mathbf{R}_8$-

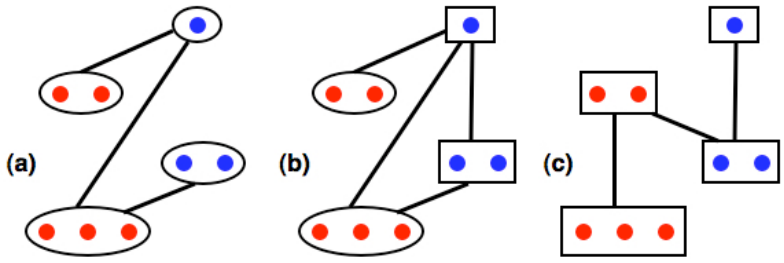

Figure 9.4. **Distinct types of connected non-trivial two-letter sequence networks.** - All three graphs are generated from the same sequence, (A, A, A, B, B, A, A, B), applying rules $\mathbf{R}_8$ (a), $\mathbf{R}_4$ (b), and $\mathbf{R}_{13}$ (c). Note the figure-background symmetry of (a) and (c): the graphs are the inverse, or complement of one another (see text). The inverse of the threshold graph (b) is also a (two-component) threshold graph, obtained from the same sequence and applying the rule $\mathbf{R}_5$ ($\mathbf{R}_4$'s complement).

graphs are typically less dense, and $\mathbf{R}_{13}$-graphs are typically denser than threshold graphs.

The connectivity rules have an additional useful interpretation as directed graphs, where the nodes represent the letters of the sequence alphabet, a directed link, e,g., from A to B indicates the rule $A \to B$, and a connection of a type to itself is denoted by a self-loop (Fig. 9.5). Because the rules are the same under permutation of types, there is no need to actually label the nodes: all graph isomorphs represent the same rule. Likewise, time-reversal symmetry means that graphs with inverted arrows are equivalent as well. Note that the direction of self-loops is irrelevant in this respect, so we simply take them as undirected.

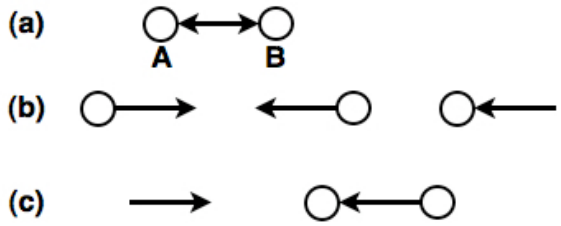

Figure 9.5. **Diagrammatic representation of rules for two-letter sequence networks.** - (a) All of the 2^2 possible connections between nodes of type A and B. (b) Three equivalent representations of the threshold rule $\mathbf{R}_4$. The second and third diagram are obtained by label permutation and time-reversal, respectively. (c) Diagrams for $\mathbf{R}_8$ and $\mathbf{R}_{13}$. Note how they complement one another to the full set of connections in part (a).

9.3.2 Alphabetical Ordering

A very special property of sequence networks is the fact that any arbitrary ensemble of such networks possesses a natural ordering, simply listing the networks alphabetically according to their sequences. In contrast, think for example of the ensemble of Erdos-Renyi random graphs of n nodes, where links are present with probability p: there is no natural way to order the 2^n graphs in the ensemble.

Plotting a structural property against the alphabetical ordering of the ensemble reveals some inner structure of the ensemble itself, yielding new insights into the nature of the nets. As an example, in Fig. 9.6 we show λ_2, the second smallest eigenvalue, for the ensemble of connected threshold nets containing $n = 8$ nodes (there are $2^7 = 128$ graphs in the ensemble, since their sequences must all start with the letter A). Notice the beautiful pattern followed by the eigenvalues plotted in this way, which resembles a fractal, or a Cayley tree: the values within the first half of the graphs in the x-axis repeat in the second half, and the pattern iterates as we zoom further into the picture.

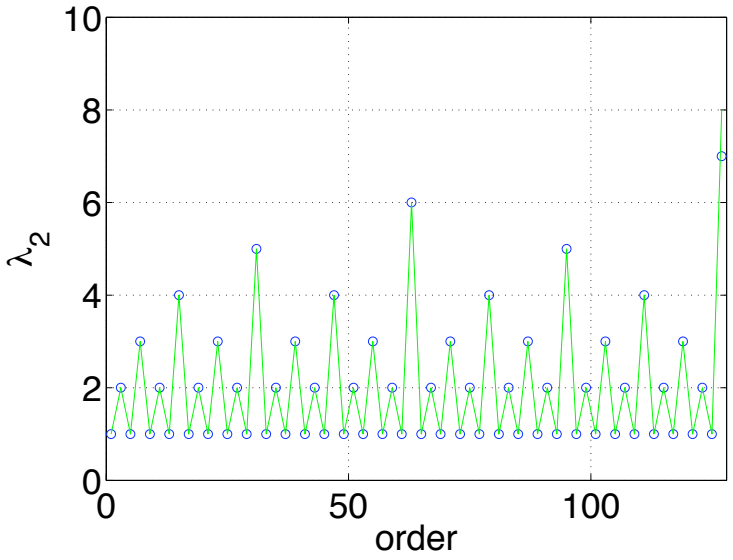

Figure 9.6. **Alphabetical ordering of threshold graphs.** - This figure shows the second smallest eigenvalues of threshold networks with $n = 8$ nodes, plotted against their alphabetical ordering.

9.4 Properties of Two-Letter Sequence Networks

Structural properties of the new classes of two-letter sequence nets, $\mathbf{R}_8$ and $\mathbf{R}_{13}$, are as easily derived as for threshold nets. Here we focus on $\mathbf{R}_8$ alone, which forms a subset of bipartite graphs. The analysis for $\mathbf{R}_{13}$ is very similar and often can be trivially obtained from the complementary symmetry of the two classes.

All connected sequence networks in the $\mathbf{R}_8$ class must begin with the letter A and end with the letter B. A sequence of this sort may be represented more compactly (HSS06) by the numbers of A's and B's in the alternating layers, $(n_{A_1}, n_{B_2}, \ldots, n_{B_l})$. We assume that there are n nodes and l layers (l is even). We also use the notation $n_A = \sum n_{A_i}$ and $n_B = \sum n_{B_i}$ for the total number of A's and B's, as well as

$$n^-_{A_j} = \sum_{i<j} n_{A_i}\,; \qquad n^+_{A_j} = \sum_{i\geq j} n_{A_i}\,, \tag{9.10}$$

and likewise for $n^\pm_{B_j}$. Finally, since all the nodes in a layer have identical properties we denote any A in the i-th layer by A_i and any B in the j-th layer by B_j. With this notation in mind we proceed to discuss several structural properties.

9.4.1 Degree, Clustering, Distance, and Betweenness

Degree: Since A's connect only to subsequent B's (and B's only to preceding A's) the degree k of the nodes is given by

$$k(A_j) = n^+_{B_j}\,; \qquad k(B_j) = n^-_{A_j}\,. \tag{9.11}$$

Clustering: There are no triangles in $\mathbf{R}_8$ nets so the clustering of all nodes is zero.

Distance: Every A is connected to the last B, so the distance between any two A's is 2. Every B is connected to the first A in the sequence, so the distance between any two B's is also 2. The distance between B_i and A_j is 1 if $j < i$ (they connect directly), and 3 if $j > i$ (B_i links to A_1, that links to B_l, that links to A_j).

Betweenness centrality: Because of the time-reversal symmetry between A and B, it suffices to analyze B nodes only. The result for A can then be obtained by simply reversing the creation sequence and permuting the letters.

The vertex betweenness $b(v)$ of a node v is defined as:

$$b(v) = \frac{1}{2} \sum_{s \neq t \neq v} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{9.12}$$

where σ_{st} is the number of shortest paths from node s to t ($s \neq t$), excluding the cases that $s = v$ or $t = v$. $\sigma_{st}(v)$ is the number of shortest paths from s to t that goes through v. The factor $\frac{1}{2}$ appears for undirected graphs since each pair is counted twice in the summation.

The betweenness of B's can be calculated from lower layers to higher layers recursively. In the first B-layer

$$b(B_2) = \frac{\frac{1}{2} n_{A_1}(n_{A_1} - 1)}{n_B}\,, \tag{9.13}$$

and

$$b(B_j) = b(B_{j-2}) + n_{A_{j-1}} \frac{\frac{1}{2}(n_{A_{j-1}} - 1) + n^-_{A_{j-1}}}{n^+_{B_j}} + n_{A_{j-1}} \frac{n^-_{B_j}}{n^+_{B_j}}, \tag{9.14}$$

for $j > 2$. The second term on the rhs accounts for the shortest paths from layer A_{j-1} to itself and all previous layers of A, and the third term corresponds to paths from A_{j-1} to B_j to A_i ($i < j-1$) to B_{j-2}. Although this recursion can be solved explicitly it is best left in this form, as it thus highlights the fact that the betweenness centrality increases from one layer to the next. In other words, the networks are *modular*, where each additional B-layer dominates all the layers below.

9.4.2 Laplacian spectrum

Laplacian spectrum: Unlike threshold networks, for $\mathbf{R}_8$ networks the eigenvalues are *not* integer, and there seems to be no easy way to compute them. Instead, we focus on the second smallest and largest eigenvalues, λ_2 and λ_n, alone, for their important dynamical role: the smaller the ratio $r \equiv \lambda_n/\lambda_2$ the more susceptible the network is to synchronization (BP02a; NM06b).

Consider first λ_2. For $\mathbf{R}_8$ it is easy to show that both the *vertex connectivity* and *edge connectivity* are equal to $\min(n_{A_1}, n_{B_l})$. Then, following an inequality in (Moh91),

$$2(1 - \cos(\frac{\pi}{n})) \min(n_{A_1}, n_{B_l}) \leq \lambda_2 \leq \min(n_{A_1}, n_{B_l})\,. \tag{9.15}$$

The upper bound seems stricter and is a reasonable approximation to λ_2 (see Fig. 9.7).

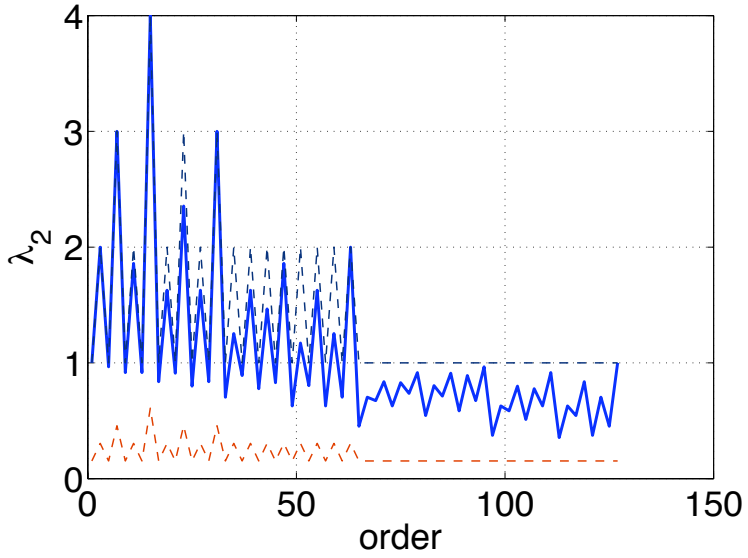

Figure 9.7. **Second smallest eigenvalues of all connected R_8 sequence networks.** - This figure shows the second smallest eigenvalues of all connected R_8 networks with $n = 8$ against their alphabetical ordering (solid curve), and their upper and lower bounds (broken lines).

For λ_n, using Theorem 2.2 of (Moh91) one can derive the bounds

$$\frac{n}{n-1}\max(n_A, n_B) \leq \lambda_n \leq n\,, \tag{9.16}$$

but they do not seems very useful, numerically. Playing with various structural properties of the networks, plotted against their alphabetical ordering, we have stumbled upon the approximation

$$\lambda_n \;\approx\; n - \left(2\frac{n_A \cdot n_B}{n} - \langle k\rangle\right)\,, \tag{9.17}$$

where $\langle k\rangle$ is the average degree of the graph, see Fig. 9.8. The approximation is exact for bipartite *complete* graphs ($l = 1$) and the relative error increases slowly with n; it is roughly at 10% for $n = 60$.

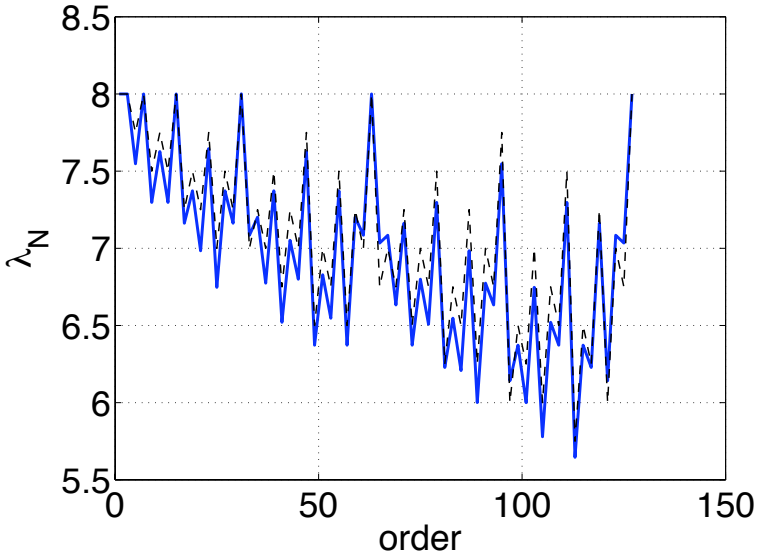

Figure 9.8. **Largest eigenvalues of all connected R_8 sequence networks.** - Plot of largest eigenvalue of all connected R_8 networks with $n = 8$ against their alphabetical ordering (solid curve), and its approximated value (broken line).

9.5 Relationship to Generalized Threshold Graphs

As discussed in the early sections of this chapter, threshold graphs can be represented by the so called creation sequence under the rule matrix $\mathbf{R}_4$ (9.5). From the classification for two-letter sequence networks, we see that besides the threshold graph, there are only 2 more types of sequence networks and they are complementary to each other. Consider, for example, creation sequences according to $\mathbf{R}_8$:

$$\mathbf{R}_4 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}. \tag{9.18}$$

One question is, is there any threshold model corresponding to this sequence network as the common threshold graph to its creation sequence? Here we propose one possible way to construct such a threshold model and show that there is a unique creation sequence under rule $\mathbf{R}_8$ correspond to this model.

The model is described in the following. Suppose the nodes are $\{1, 2, ..., n\}$ with weights

$$0 \leq x_1 \leq x_2 \leq ... \leq x_n < 2\theta. \tag{9.19}$$

where θ is the prescribed threshold.

The connection between nodes i and j is determined by:

$$i \text{ connects to } j \text{ iff } |x_i - x_j| > \theta \tag{9.20}$$

so that given the weights and the threshold, we can generate the corresponding graph. To avoid confusion with the usual threshold graph, we call this type of graph to be θ^--graph, and the usual threshold graph will be classified as θ^+-graph. To see how the same graph can be obtained by the creation sequence under $\mathbf{R}_8$, we first classify the nodes into two types A and B according to their weights:

$$i \in A \text{ iff } 0 \leq x_i < \theta; \text{ otherwise } i \in B. \tag{9.21}$$

Now for convenience, rewrite the weights of the nodes to be

$$0 \leq u_1 \leq u_2 \leq ... \leq u_{n_A} \leq \theta < v_1 \leq v_2 \leq ... \leq v_{n_B} < 2\theta \tag{9.22}$$

where the first n_A weights correspond to A nodes and the rest to B nodes, and $n_A + n_B = n$.

To obtain a creation sequence of a threshold graph of this type under rule R_8, we proceed as follows:

- Let $S = \{s_1, s_2, ..., s_n\}$ be a creation sequence, with $s_k = A$ or B.
- Set $i = 1, j = 1$.
- For $k = 1, 2, ..., n$, do:
  If $|x_i - x_j| > \theta$
  Set $S_k = A$ and $i = i + 1$;
  Otherwise
  Set $S_k = B$ and $j = j + 1$.
- End.

It is understood that if the u_i are exhausted before the end of the loop, the remainding B nodes are automatically affixed to the end of the sequence (and similarly for the v_j).

For example, the $\mathbf{R}_8$ sequence network (A, A, A, B, B, A, A, B) corresponds to a θ^--graph with weights on nodes $\{0.1, 0.2, 0.3, 0.5, 0.7, 1.6, 1.7, 2\}$ and connection threshold $\theta = 1.2$. And we can obtain the creation sequence from the weights and threshold using the above algorithm, as you may check.

Consider now the converse problem: given a graph created from the sequence $(s_1, s_2, \ldots, s_N)$ with the rule $\mathbf{R}_8$, we derive a (non-unique) set of weights $\{x_i\}$ such that connecting any two nodes with $|x_i - x_j| > \theta$ results in the same graph. Rewrite first the creation sequence into its compact form $(N_{A_1}, N_{B_2}, ..., N_{B_l})$, and assign weights p for nodes A in layer p, weights $n+q$ for nodes B in layer q, and set the threshold at $\theta = n$. For example, the sequence (A, A, A, B, B, A, A, B) has a compact representation $(3, 2, 2, 1)$, with $l = 4$ layers, so the three A's in layer 1 have weights 1, the two B's in layer 2 have weights 6, the two A's in layer 3 have weights 3, and the single B in layer 4 has weight 8. The weights $\{1, 1, 1, 6, 6, 3, 3, 8\}$, with connection threshold $\theta = 4$, reproduce the original graph.

Sequence networks obtained from the rule $\mathbf{R}_{13}$ can be also mapped to difference threshold graphs in exactly the same way, only that the criterion for connecting two nodes is then $|x_i - x_j| < \theta$, instead of $|x_i - x_j| > \theta$, as for $\mathbf{R}_8$. This type of networks may be a particularly good model for social networks, where the weights might measure political leaning, economical status, number of offspring, etc., and agents tend to associate when they are closer in these measures.

The mapping of sequence networks to generalized threshold graphs may be helpful in the analysis of some of their properties, for example, for finding the *isoperimetric number* of a sequence graph (Moh91).

9.6 Discussion and Open Problems

In this chapter we have introduced a new class of networked called sequence networks, obtained from a sequence of letters and fixed rules of connectivity. We have classified all the two-letter sequence networks, which contain threshold networks, and in addition two newly discovered classes. The classification for three-letter sequence networks is not as ease, and can be found in (SNb08).

By making use of the layer representation and alphabetical ordering, structural properties of the newly discovered two-letter sequence networks have been analyzed.

The diameter of sequence networks grows linearly with the number of letters in the alphabet and for a three-letter alphabet it is already 3 or 4, comparable to many everyday life complex networks. Realistic diameters might be achieved with a modest expansion of the alphabet.

There remain numerous open questions: Applying symmetry arguments we have managed to reduce the class of 2-letter networks to just 3 types and 3-letter networks to just 30 types (SNb08), but we have not ruled out the possibility that some overlooked symmetry might reduce the list further. How to classify higher order (i.e., with more letters) sequence networks other than the brute-force approach is still unclear.

The question of which sequences lead to connected networks can be studied by inspection for small alphabets, but we have no comprehensive approach to solve the problem in general. We have shown how to map sequence networks to generalized types of threshold nets, in some cases — Is such a mapping always possible? Is there a systematic way to find such mappings for any sequence rule? What kinds of networks would result if the connectivity rules applied only to the q (where q is finite) preceding letters, instead of to *all* preceding letters, or replacing the deterministic connectivity rule by some *probabilistic* process?

Chapter 10

Greedy Connectivity of Embedded Networks

10.1 Geographical Graphs

For some networks there are underlying structural constraints other than the explicit connections between nodes. For example, a social network is often influenced by geographical locations: people in the same neighborhood or who go to work at the same place are more likely to be friends. In fact, the study of how mail/message can be sent from a person to a given randomly chosen target (Mil67; DMW03) shows that most of the time people choose to send mails/messages to friends who live geographically closer to the target.

The idea of embedding a network in some metric space has been proposed to construct scale-free networks in lattices (RCb02), to explore local navigability of complex networks (BKC09), and so on. The key advantage of having some knowledge of an underlying metric space for the nodes is to allow efficient navigation without global coordination (Kle00a; Kle00b; CCSb09; CD09).

Motivated by such applications, in this chapter we define a geographical graph to be a graph with additional metric structure on the set of edges, and study the

so called *greedy connectedness* of some benchmark network models such as lattice, Erdos-Renyi networks, and Watts-Strogatz like networks. The greedy connectedness (to be defined in the following section) takes into the account the fact that when global information is not available, a node may not able to find its path to some target, even if there indeed exists one. We found that this concept is best explained by a real world example shown in Fig. 10.1.

Figure 10.1. **Blenheim Palace Garden Maze.** - A maze can be thought of a connected graph. However, can everyone finds his way out, even if we know the geographical location of the exit?

When nodes in a graph are embedded in a coordinate space, the metric defined on the edges is simply induced by the distance between points/vectors in the corresponding coordinate space.

Definition 10.1 (Geographical graph) *A* ***geographical graph*** $\tilde{G} = (V, E; \Phi)$ *is a graph* $G = (V, E)$ *with additional geographical structure which defines a metric space* Φ *for the set of nodes* V*. The metric will be denoted by* ϕ*, which is a function*

$$\begin{aligned} \phi : V \times V &\rightarrow \mathbb{R}^+ \cup \{0\} \\ (i, j) &\mapsto \phi(i, j). \end{aligned}$$

Thus, by the properties of metric, we have: $\phi(i,i) = 0$; $\phi(i,j) = \phi(j,i) > 0$ whenever $i \neq j$; and $\phi(i,j) \leq \phi(i,k) + \phi(k,j)$ for any $i, j, k \in V$.

10.2 Greedy Connectivity: Definition and Properties

10.2.1 Path and Connectivity in Graphs

Paths and shortest paths play important role in graph theory, and also in many applications as discussed in Chapter 7. In this subsection we review some definition and results regarding paths. Much of the results (although stated slightly differently) can be found in many classical graph theory books, for example, (Die06; Wes00).

First let us define a path in a graph $G = (V, E)$ (here and in the following, this notation of a graph will be adopted without special declaration).

Definition 10.2 (Path) *A sequence of numbers $p = (i_1, \ldots, i_{\ell+1})$ is a **path** of length ℓ (the path length is simply $|p| - 1$) if*

$$\{i_1, \ldots, i_{\ell+1}\} \subset V, \tag{10.1}$$

and for each $k \in \{1, \ldots, \ell\}$,

$$(i_k, i_{k+1}) \in E. \tag{10.2}$$

*Here the nodes i_1 and $i_{\ell+1}$ are called **endpoints** of the path.*

Lemma 10.1 *A sub-path of a path is also a path. On the other hand, given two paths $(i_1, \ldots, i_{\ell+1})$ and $(j_1, \ldots, j_k)$, if $i_{\ell+1} = j_1$, then the sequence of nodes $(i_1, \ldots, i_{\ell+1}, j_2, \ldots, j_k)$ also forms a path.*

Note that the nodes appearing in a path can repeat. Thus, a path can contain loop(s). The following lemma is based on the observation that by repeating a path (or think of 'gluing' two paths together), one obtains a longer path between two nodes.

Lemma 10.2 (Number of Paths between Connected Nodes) *In a given graph, suppose there is one path between two nodes s and t, then there are infinitely many path with different lengths between the same pair of nodes s and t.*

A path in a graph can be thought of as a spatial trajectory from one node to another. This trajectory can be infinitely long, which contains repeated patterns of nodes which reminds us of periodic orbits from dynamical systems. Indeed, the relationship between dynamical systems and graphs can be explored through *symbolization* (MH38), usually accomplished through *generating partitions* (PCB06), another exciting concept in dynamical systems.

Definition 10.3 (Shortest Path) *Let P_{st} denote the set of all paths with endpoints s and t. The **shortest path** connecting nodes s and t is a path $p_m \in P_{st}$ such that $\forall p \in P_{st}$,*

$$|p_m| \leq |p|. \tag{10.3}$$

The length of a shortest path between s and t is then $|p_m| - 1$, and will be denoted by d_{st}. The shortest path length is also referred to simply as path length for convenience.

In a general graph, from one node s to another node t there may be multiple shortest paths with the same path length. However, in a *tree* (a tree is a graph with no loops), all shortest paths are unique. The following lemma is also useful in some proofs relating shortest paths.

Lemma 10.3 *A sub-path of a shortest path is also a shortest path.*

Thus, if $(i_1, \ldots, i_{\ell+1})$ is a shortest path, then $\forall j, k \in \{1, \ldots, \ell + 1\}$ where $j < k$, the path $(i_j, \ldots, i_k)$ is also a shortest path (connecting the nodes i_j and i_k).

Remark 10.1 *Any node appearing in a shortest path can only appear once, since otherwise, by the above lemma, we may obtain another shortest path with smaller path length.*

Definition 10.4 (Connectedness of Nodes) *In an undirected graph, two nodes s and t in a graph are called* ***connected*** *if there is a path starting at s and end at t. Connectedness defines an equivalence relationship on the set of nodes. A* ***component*** *in a graph is a maximal connected subgraph.*

If a graph has more than one components, then there exists pair of nodes which cannot be reached from each other using edges in the graph. This motivates the definition of connectedness of a graph.

Definition 10.5 (Connectedness of a Graph) *A graph $G = (V, E)$ is* ***connected*** *if there is a unique component (equals the graph itself).*

In the next subsection we shall explore some properties of greedy paths in graphs, and compare them especially with the properties of paths and shortest paths.

10.2.2 Greedy Paths and Greedy Connectivity

Definition 10.6 (Greedy Path) *In a geographical graph $\tilde{G} = (V, E; \Phi)$, a path $p = (i_1, \ldots, i_{\ell+1})$ is a* ***greedy path*** *from node i_1 to node $i_{\ell+1}$ of length ℓ if For each $k \in \{1, \ldots, \ell\}$,*

$$\phi(i_{k+1}, i_{\ell+1}) \leq \phi(j, i_{\ell+1}) \quad \forall j \text{ s.t. } (i_k, j) \in E. \tag{10.4}$$

Remark 10.2 *By definition, a greedy path is automatically a path. The converse is not true.*

Remark 10.3 (Symmetry Breaking of Greedy Path) *For a graph that is undirected, unlike paths/shortest paths, a greedy path from a node s to t does not guarantee the existence of greedy path going backwards (from t to s).*

Remark 10.4 (Transitivity Breaking of Greedy Path) *Unlike paths/shortest paths, a path $(i_1, \ldots, i_\ell, j_1, \ldots, j_k)$ may not be a greedy path even if both $(i_1, \ldots, i_\ell)$ and $(j_1, \ldots, j_k)$ are.*

Definition 10.7 (Greedy Connectedness) *A geographical graph is* ***greedily connected*** *if* $\forall (i,j) \in V \times V$*, there is a greedy path* $p = (i_1, \ldots, i_{\ell+1})$ *with* $i_1 = i$ *and* $i_{\ell+1} = j$*.*

Due to the above two lemma, even in an undirected geographical graph, there is no well defined components induced by the greedy connectedness. Next we explore the relationship between connectedness and greedy connectedness.

Lemma 10.4 (Greedy connectivity and connectivity) *Greedy connectedness* $\Rightarrow$ *connectedness. More precisely, if a geographical graph is greedily connected, then it is also connected.*

Remark 10.5 *Connectedness does not necessarily imply greedy connectedness.*

Lemma 10.5 (Sub-paths of a greedy path) *Suppose* $p = (i_1, \ldots, i_{\ell+1})$ *is a greedy path, then* $\forall k \in \{1, \ldots, \ell\}$*, the path* $(i_k, \ldots, i_{\ell+1})$ *is also a greedy path.*

Remark 10.6 *Unlike in the case of paths, a sub-path of a greedy path may not be a greedy path.*

Remark 10.7 *For a greedily connected graph, adding edges does not guarantee greedy connectedness of the graph.*

Lemma 10.6 (Uniqueness of greedy paths) *Let* $\tilde{G} = (V, E; \Phi)$ *be a geographical graph. Suppose that* Φ *is a* $1 - to - 1$ *function on* E*, i.e.,* $\forall (i,j), (i',j') \in E$*,*

$$\phi(i,j) = \phi(i',j') \Rightarrow (i,j) = (i',j'), \tag{10.5}$$

then all greedy paths in the graph are unique.

Remark 10.8 *The uniqueness property is quite general. For example, if we embed the set V in $\mathbb{R}^n$ where points (coordinates for nodes) are drawn from a continuous distribution, then the condition for uniqueness holds with high probability. Thus, greedy paths are generically unique with respect to the choice of embedding of nodes into a metric space. To be more rigorous, we need assumptions about the space of Φ under consideration.*

10.2.3 Probabilistic Paths

In this subsection and following, we will use the word path to mean either a shortest path or greedy path, rather than general paths. However, when it comes to special circumstances, we shall differ between whether a path is a shortest path or a greedy path, to avoid confusion.

Define first the quantity $P(\ell, m)$ by

$$\begin{aligned} P(\ell, m) &\equiv \text{probability that two sites of geographical distance } m \\ &\quad \text{is connected through a greedy path of length } \ell. \end{aligned} \tag{10.6}$$

In practice, there are various factors causing failures when we attempt to follow a long path to send a message for example. These factors can be taken into account by a simple but yet crucial analytical treatment.

Suppose that for each edge in a path there is a constant probability of failure, defined by $e^{-\mu}$ for a given $\mu \geq 0$, then for long paths this factor accumulates in an exponential sense whenever $\mu > 0$. This key feature captures the fact that the longer a path is in practice, the less possible we will be able to follow the whole path.

Therefore, we define, for a given μ, the following quantity:

$$\begin{aligned} P_\mu(m) &\equiv \sum_\ell P(\ell, m) e^{-\mu \ell} \\ &\equiv \text{probability that two sites of geographical distance } m \\ &\quad \text{is connected under success rate } e^{-\mu}. \end{aligned} \tag{10.7}$$

The overall connectivity of a graph under the failure rate $e^{-\mu}$ is then assessed by

the quantity

$$\langle P_\mu \rangle = \frac{\sum_m N_m \cdot P_\mu(m)}{\sum_m N_m} \tag{10.8}$$

where N_m is the number of pairs of sites at geographical distance m.

10.3 Greedy Connectivity: Computation

In this section we assume that the geographical graph satisfies the condition Eq. (10.5), so that all greedy paths are unique.

10.3.1 Brute-Force Approach

For a given geographical graph $\tilde{G} = (V, E; \Phi)$, it is obvious how to find a greedy path from one node to the other by simply following the definition of a greedy path. The approach is presented in pseudo code below:

Brute-Force Approach for Finding a Greedy Path between Two Nodes

---

Given: $\tilde{G} = (V, E; \Phi)$, a geographical graph.
Goal: Find a greedy path between nodes s and t.

Let: $p = (s)$ be a vector to contain the path.
Let: $i = s$ and $stop = 0$.
while $(i \neq t)$ & $(stop = 0)$ **do**
  Find $j \in E$ s.t., $\phi(j,t) < \phi(k,t)$ for all $k \neq j$ and $k \in E$
  if $\phi(j,t) > \phi(i,t)$ **then**
    Set $stop = 1$.
  else
    Add node j to the end of the sequence p and set $i = j$.
  end if
end while
if i=t **then**

Output p as the greedy path.

else

No greedy path found.

end if

The brute-force approach is simple to implement and understand, but it costs $O(|E|)$ operations to find a path between any pair of nodes. Thus, to find all greedy paths in a graph we need to run the algorithm $|V|^2$ times. The total number of operations needed would thus be $O(|V| \cdot |E|^2)$. For a sparse, connected graph where $|E| = O(|V|)$, the number of operations simplifies to $O(|V|^3)$.

In the next subsection we will show that there exists a much faster way to compute all the greedy paths in a geographical graph, which only requires $O(|V| \cdot |E| + |V|^2)$ operations, or $O(|V|^2)$ when the graph is sparse.

10.3.2 Geographical Breadth-First-Search (BFS) for Greedy Paths

In this subsection we show that the popular Breadth-First-Search for finding usual shortest paths can be used to find all greedy paths *to* a given node in a geographical graph. The approach is called *geographical BFS*.

In the following we provide Matlab code for computing all greedy paths to a target node t in a geographical graph. The idea is to perform a breadth-first-search like algorithm starting from the target node, explore from closer to farther away in the geographical distance sense all the other nodes to check the existence of greedy paths to the target.

We have assumed that any greedy path, if exist, is always unique. In practice, for circularly embedded graphs such as lattices, this can be achieved by perturbing the position of each node by some small amount.

Modified Breadth-Force Search for All Greedy Paths to a Given Node

---

---

function [d,p] = GreedySearch(A,X,t)

%% **Assumptions:**

% A is unweighted (& undirected?)

%% **Breadth-first-search in the Geographical induced Metric**

% A: adjacency matrix (A(i,j)==1 iff there is an edge from i to j)

% X: coordinate mapping

% X(:,i) (i-th column of X) is the coordinate of node i

% t: targeting node

%% **Edges in the graph**

n = length(A);

A = sparse(A);

Nbs = cell(n,1);

for i = 1 : n

  Nbs{i} = find(A(:,i));

end

%% **Initial set up**

p = -ones(1,n); % p(i) is the next node in the greedy path from i to t

d = -ones(1,n); % d(i) is the greedy path length from i to t

burn = zeros(1,n); % burn(i) = 1 iff i has been explored

%% **Geographical distances to the target t (using Euclidean norm)**

phi = X - kron(ones(1,n),X(:,t)); % phi(i) is the geo.dist to t

phi = sqrt(sum(phi.*phi)); % distance to the target

[B,IX] = sort(phi); % sorting from closer to farther away

% IX(1) is the index of the node who is closest to t (i.e., t)

% IX(2) is the index of the second closest node to t

% (...)

```
%% Before search
d(t) = 0;
p(t) = 0;
burn(t) = 1;
```

```
%% Start search from closer to farther away to the target t
for i = 1 : n-1
    k = IX(i);
    if burn(k)==0
        burn(k) = 1;
    end
    for indj = 1 : length(Nbs{k})
        j = Nbs{k}(indj);
        if burn(j)==0
            if d(k)>=0
                d(j) = d(k) + 1;
                p(j) = k;
            end
            burn(j) = 1;
        end
    end
end
%%
```

This algorithm costs as much as the usual BFS, which requires $O(|V|+|E|)$ operations. To find all greedy paths, we simply need to run the algorithm $|V|$ times. Thus, the problem of finding greedy connectedness of an unweighted geographical graph $\tilde{G} = (V, E; \Phi)$ can be solved computationally in $O(|V| \cdot |E| + |V|^2)$ operations; or $O(|V|^2)$ when the graph is sparse.

10.4 Greedy Connectivity for some Network Models

In this section we explore numerically the greedy connectivity of some network models under different success rates (a concept introduced in Section 10.2.3).

10.4.1 Circular Embedding

We will study a specific type of embedding called circular embedding. This embedding simply embeds the space of nodes on a circle in $\mathbb{R}^2$. To be precise, if $V = \{1, \ldots, n\}$, then node i will be put at position

$$x_i = \Big(\cos(\frac{i}{n}2\pi), \sin(\frac{i}{n})2\pi) \Big) \in \mathbb{R}^2, \tag{10.9}$$

and ϕ can be defined either as $\phi(i,j) = \min(|i-j| \mod n,\ |j-i| \mod n)$ or simply $\phi(i,j) = ||x_i - x_j||$ where $||.||$ is the usual Euclidean distance; both definition will yield the same results for greedy paths, since they are *isomorphic* metrics.

For a circularly embedded graph of $2L$ nodes, the range of geographical distance m is from 1 to L, whereas the range of greedy path length ℓ is also from 1 to L. N_m is always $4L$. So for circularly embedded graphs,

$$P_\mu(m) \equiv \sum_{\ell=1}^{L} P(\ell, m) e^{-\mu\ell} \tag{10.10}$$

and

$$\langle P_\mu \rangle \equiv \frac{\sum_m N_m \cdot P_\mu(m)}{\sum_m N_m} = \frac{\sum_{m=1}^{L} 4L \cdot P_\mu(m)}{\sum_{m=1}^{L} 4L} = \frac{1}{L} \sum_{m=1}^{L} P_\mu(m). \tag{10.11}$$

10.4.2 Circularly Embedded Lattices

A lattice $G_L(L,d)$ is a circular graph where there are $2L$ nodes, each connect to d of its nearest neighbors at each side. A circularly embedded lattice $\tilde{G}_L(L,d)$ is a lattice $G_L(L,d)$ embedded as Eq.(10.9).

The equation for $P(\ell, m)$ in this case reads

$$P(\ell, m) = \delta_{\ell, \lceil \frac{m}{d} \rceil} \tag{10.12}$$

where $\lceil x \rceil$ is the least integer which is greater or equal to x. Then, for $P_\mu(m)$, we have

$$P_\mu(m) \equiv \sum_{\ell=1}^{L} P(\ell, m) e^{-\mu\ell} = \sum_{\ell=1}^{L} \delta_{\ell, \lceil \frac{m}{d} \rceil} e^{-\mu\ell} = e^{-\mu \lceil \frac{m}{d} \rceil}. \tag{10.13}$$

Finally,

$$\begin{aligned}
\langle P_\mu \rangle &= \frac{1}{L} \sum_{m=1}^{L} P_\mu(m) \\
&= \frac{1}{L} \Big[\sum_{m=1}^{d} e^{-\mu} + \sum_{m=d+1}^{2d} e^{-2\mu} + \cdots + \sum_{m=(\lfloor \frac{L}{d} \rfloor - 1)d+1}^{\lfloor \frac{L}{d} \rfloor d} e^{-\lfloor \frac{L}{d} \rfloor \mu} + \sum_{m=\lfloor \frac{L}{d} \rfloor d+1}^{L} e^{-(\lfloor \frac{L}{d} \rfloor + 1)\mu} \Big] \\
&= \frac{d}{L} \Big[e^{-\mu} + e^{-2\mu} + \cdots + e^{-\lfloor \frac{L}{d} \rfloor \mu} + \frac{L - \lfloor \frac{L}{d} \rfloor d}{d} e^{-(\lfloor \frac{L}{d} \rfloor + 1)\mu} \Big].
\end{aligned} \tag{10.14}$$

Assuming for convenience that $\frac{L}{d}$ is an integer, then the above equation can be further simplified as:

$$\langle P_\mu \rangle = \langle P_\mu \rangle = \frac{d}{L} \sum_{k=1}^{\frac{L}{d}} e^{-k\mu} = \begin{cases} \frac{d}{L} e^{-\mu} \frac{1 - e^{-\frac{L}{d}\mu}}{1 - e^{-\mu}}, & \text{if } 0 < e^{-\mu} < 1; \\ 1, & \text{if } e^{-\mu} = 1. \end{cases} \tag{10.15}$$

Fig. 10.2 shows numerically simulated $\langle P_\mu \rangle$ compared with theory given by Eq. (10.15).

10.4.3 Circularly Embedded Random Graphs

A random graph $G_R(L, d)$ is a graph where there are $2L$ nodes, and every pair of nodes are connected by an edge with probability $\frac{d}{L}$. A circularly embedded random graph $\tilde{G}_R(L, d)$ is a random graph $G_R(L, d)$ embedded as Eq.(10.9).

For this model, the *master equation* for $P(\ell, m)$ can be derived as, for $1 \leq \ell, m \leq$

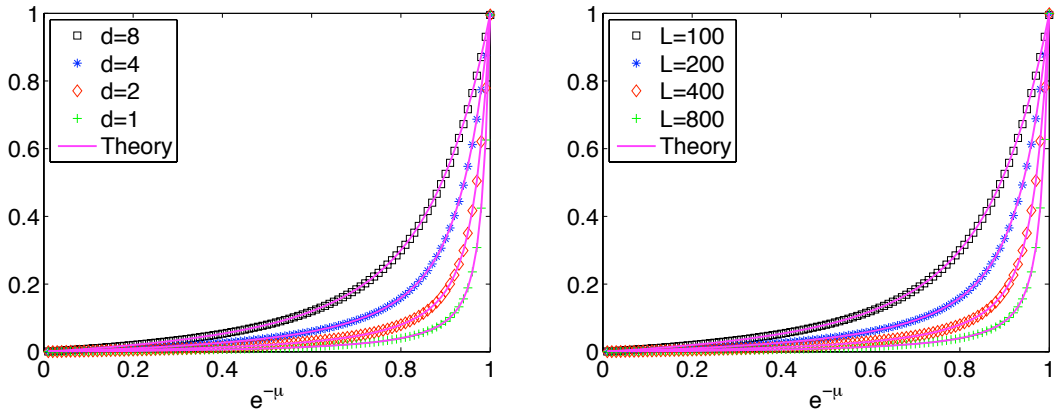

Figure 10.2. **Greedy connectedness of circular lattices.** - Left panel shows simulation and theoretical results of $\langle P_\mu \rangle$ according to Eq. (10.15) for circularly embedded lattices of fixing $L = 100$; right panel shows $\langle P_\mu \rangle$ for lattices of fixed $d = 8$.

L:

$$\begin{aligned} P(\ell, m) &= \lambda P(\ell-1, 0) + (1-\lambda)\omega P(\ell-1, 1) + (1-\lambda)^3 \omega P(\ell-1, 2) \\ &\quad + \cdots + (1-\lambda)^{2m-3}\omega P(\ell-1, m-1) \\ &= \lambda P(\ell-1, 0) + \omega \sum_{k=1}^{m-1} (1-\lambda)^{2k-1} P(\ell-1, k). \end{aligned} \tag{10.16}$$

Here

$$\lambda \equiv \frac{d}{L}, \quad \omega \equiv 1 - (1-\lambda)^2 = \lambda(2-\lambda). \tag{10.17}$$

The boundary conditions for this equation are:

$$\begin{cases} P(0,0) = 1; \\ P(0,m) = 0, \quad \text{if } m \geq 1; \\ P(\ell, m) = 0, \quad \text{if } \ell > m. \end{cases} \tag{10.18}$$

Next we use the above master equation to derive a recurrence equations on between

$P_\mu(m)$ as follows:

$$\begin{aligned}
P_\mu(m) &\equiv \sum_{\ell=1}^{L} P(\ell,m)e^{-\mu\ell} = \sum_{\ell=1}^{m} P(\ell,m)e^{-\mu\ell} \\
&= \lambda\sum_{\ell=1}^{m} P(\ell-1,0)e^{-\mu\ell} + \omega\sum_{\ell=1}^{m}\sum_{k=1}^{m-1}(1-\lambda)^{2k-1}P(\ell-1,k)e^{-\mu\ell} \\
&= \lambda e^{-\mu} + \omega\sum_{k=1}^{m-1}(1-\lambda)^{2k-1}\sum_{\ell=1}^{m} P(\ell-1,k)e^{-\mu\ell} \\
&= \lambda e^{-\mu} + \omega\sum_{k=1}^{m-1}(1-\lambda)^{2k-1}e^{-\mu}\sum_{\ell=1}^{k+1} P(\ell-1,k)e^{-\mu(\ell-1)} \\
&= \lambda e^{-\mu} + \omega\sum_{k=1}^{m-1}(1-\lambda)^{2k-1}e^{-\mu}P_\mu(k).
\end{aligned}$$

That is, for $2 \le m \le L$:

$$P_\mu(m) = e^{-\mu}\Big[\lambda + \omega\sum_{k=1}^{m-1}(1-\lambda)^{2k-1}P_\mu(k)\Big].$$

Thus, we have the recurrence equations for $P_\mu(m)$, as:

$$\begin{aligned}
& P_\mu(m) - P_\mu(m-1) = e^{-\mu}\omega(1-\lambda)^{2m-3}P_\mu(m-1) \\
\Rightarrow\ & P_\mu(m) = \Big[1 + e^{-\mu}\omega(1-\lambda)^{2m-3}\Big]P_\mu(m-1). \qquad (10.19)
\end{aligned}$$

The initial condition $P_\mu(1)$ can be computed as

$$P_\mu(1) = P(1,1)e^{-\mu\ell} = \lambda e^{-\mu\ell}. \qquad (10.20)$$

Therefore, the general formula for $P_\mu(m)$ can be obtained, as:

$$P_\mu(m) = \lambda e^{-\mu}\prod_{k=2}^{m}\Big[1 + e^{-\mu}\omega(1-\lambda)^{2k-3}\Big]. \qquad (10.21)$$

Finally,

$$\begin{aligned}
\langle P_\mu\rangle &= \frac{1}{L}\sum_{m=1}^{L} P_\mu(m) \\
&= \lambda e^{-\mu}\frac{1}{L}\sum_{m=1}^{L}\prod_{k=2}^{m}\Big[1 + e^{-\mu}\omega(1-\lambda)^{2k-3}\Big]. \qquad (10.22)
\end{aligned}$$

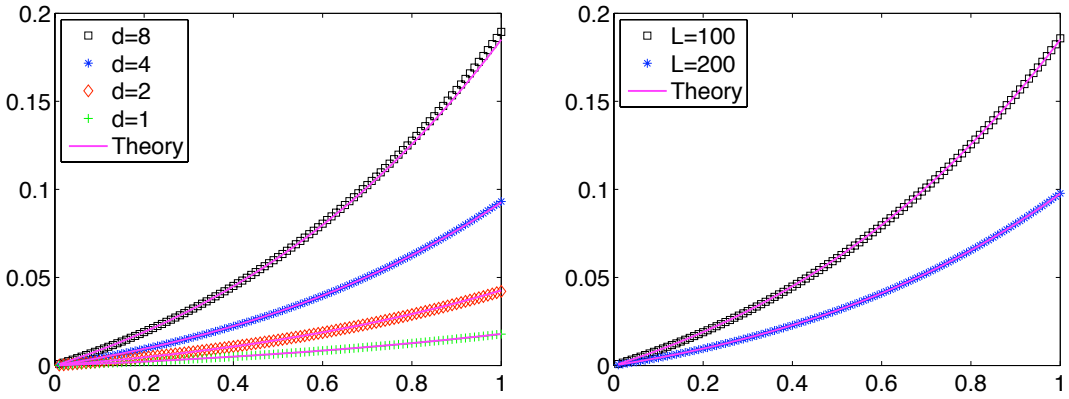

Figure 10.3. **Greedy connectedness of circularly embedded random graphs.** - Left panel shows simulation and theoretical results of $\langle P_\mu \rangle$ according to Eq. (10.22) for circularly embedded random graphs of fixing $L = 100$; right panel shows $\langle P_\mu \rangle$ for such graphs of fixed $d = 8$. Horizontal axis is the value of $e^{-\mu}$. All numerical curves are averaged over 100 random realizations.

10.4.4 Lattice Rewiring Model: the Interplay between Short and Long Range Connections

In the previous two subsections we have numerically shown that a random graph always has worse greedy connectedness than a lattice, when both are circularly embedded. It seems that randomness would cause the decrease of greedy connectedness in general. Is this true?

In the following we show that although complete randomness might be the worst case for greedy connectedness, as shown in the example of a Erdos-Renyi graph, an appropriate level of randomness can indeed improve the greedy connectedness significantly. The example is shown in Fig. 10.4.

The increasing parts of the two curves at small ϵ region where $\mu > 0$ indicates the importance of randomness in an ordered world to allow efficient local navigation. The fact that there is always a *unique* maximum of the curve whenver $\mu > 0$ is also interesting, and can be analyzed theoretically by means of master equations on $\langle P(\mu) \rangle$ (CCSb09), but will not be discussed in this thesis.

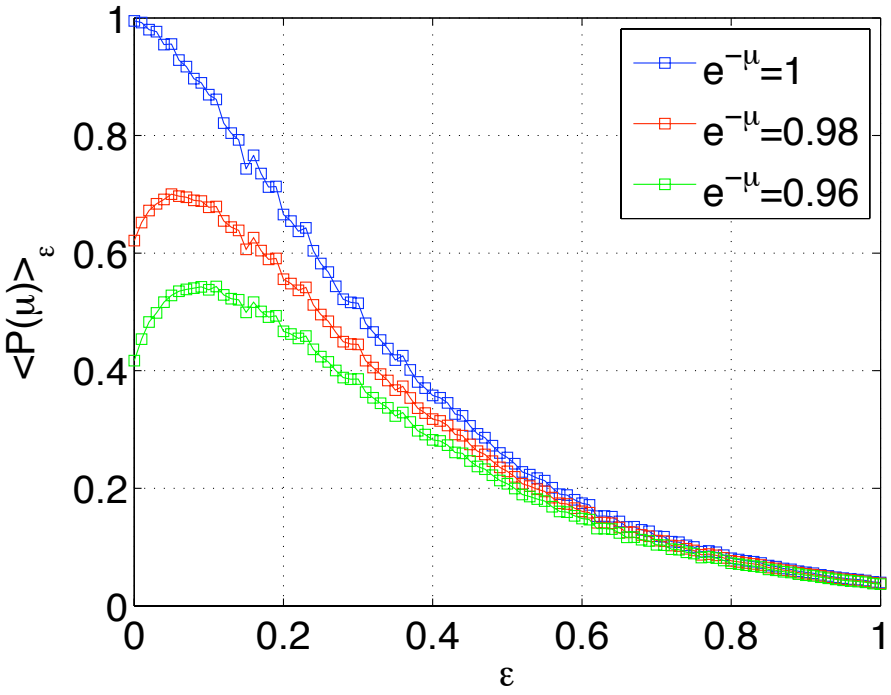

Figure 10.4. **Greedy connectedness of small world graphs.** - Here we start with a circular lattice of $n = 200$ nodes and $k = 4$. For given ϵ (horizontal axis), we rewire each edge in the lattice with probability ϵ. The vertical axis corresponds to $\langle P(\mu) \rangle$ as defined in Eq. (10.8). Here three curves are shown for different choices of μ. The plotted curves are average over 20 different realizations.

10.5 Discussion and Open Problems

In this chapter we introduce the concept of greedy connectedness of geographical graphs. We have shown a few interesting properties of greedy paths and greedy connectedness which are different from those of usual paths and connectedness. Furthermore, we introduce the concept of failure rate for paths in graphs, which can be used to model communications in realistic situations. To compute greedy paths, we develop an efficient algorithm based on classical BFS. Numerical study of the greedy connectedness of a few network models are also shown for circularly embedded lattices, random networks, and Watts-Strogatz networks. We have found that there is a maximum of the greedy connectedness in Watts-Strogatz networks for any non zero failure rate, and this maximum usually takes place at a small rewriting probability, which seems to be realistic.

Despite its theoretical relevance, the application to practical problems is promising: a deeper understanding of what can improve/worsen greedy connectedness in a more rigorous sense is useful for the design of communication systems.

Chapter 11

Conclusion

In this thesis we have focused on several different aspects of the general area of complex systems modeling. The three main areas we have studied are: dynamics on networks, dynamics of networks, and network modeling.

(1) *Dynamics on networks.*

In this area we have studied an important problem about synchronization in systems of coupled chaotic oscillators where the oscillators were not necessarily governed by the same dynamics, a big assumption usually made in the area. We developed a generalized master stability function approach to analyzing the stability of synchronization in this case (Chapter 2) and (SBN09d; SBN09c). Furthermore, an important question regarding how to reduce the complexity of a spatio-temporal system was raised and analyzed in Chapter 3 and (SBN09b). The difficulty of judging the quality of a reduced order model arises when the individual components of the systems are governed by chaotic oscillators. In Chapter 4 we reviewed a concept called shadowing and developed new analysis about how optimal shadowing can be used to measure the quality of model for chaotic systems. Such techniques were used in Chapter 5 for the study of model reduction of coupled chaotic oscillators (SBN09b).

(2) *Dynamics of networks.*

Real world networks change from time to time. However, the majority of the research in this area has been focusing on network properties that are static. Part of

the difficulty of modeling a time dependent network comes from the fact that computing network statistics for a time varying graph topology is costly by traditional methods. Motivated by such problems, we developed efficient algorithms based on update formulae of network statistics to track the evolution of large, evolving networks. Those statistics included local statistics such as degree, clustering coefficient, assortativity coefficient, and modularity (Chapter 6) and (SBBS09); and global statistics such as shortest paths (SBB$^+$09) and spectral radius of adjacency matrices (MSN09) (Chapter 7).

(3) Problems related to *network modeling*.

Various problems have attracted us. In Chapter 8 we proposed a method based on symmetry reduction to efficiently represent a graph, aiming at the optimal compression of graphical structures, a seemingly simple, but yet unsolved problem (SBb08). In Chapter 9 we proposed a new class of networks called sequence networks (SNb08), which are networks generated by deterministic rules applying to a sequence of numbers. Those networks have beautiful analytical properties such as low storage requirements and fast computations, as well as the advantage of being easy to design. In Chapter 10 (and (Sb09)) we introduced the concept of geographical graphs to model graphs that are embedded in some metric spaces, and define a new type of connectedness of such graphs, called greedy connectedness. Greedy connectedness under finite failure rate captures the key feature that in real life, paths are not perfectly robust. Numerical study in a few examples suggested that an ordered network with some randomness is optimal for such situations.

Conducted works that were not discussed in the thesis include (SGL07), (SBN09a), and (SBPD09).

References

[AB02] R. Albert and A.-L. Barabasi. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47, 2002.

[ADK+08] A. Arenas, A. Diaz-Guilera, J. Kurths, Y. Moreno, and C. Zhou. Synchronization in complex networks. *Physics Reports*, 469:93–153, 2008.

[ADP06] A. Arenas, A. Diaz-Guilera, and C. J. Perez-Vicente. Synchronization reveals topological scales in complex networks. *Phys. Rev. Lett.*, 96:114102, 2006.

[Ano67] D. V. Anosov. Geodesic flows on closed riemannian manifolds with negative curvature. *Proc. Steklov Inst. Math.*, 90, 1967.

[BA99] A.-L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509, 1999.

[BBSb08] J. P. Bagrow, E. M. Bollt, J. D. Skufca, and D. ben-Avraham. Portraits of complex networks. *Euro. Phys. Lett.*, 81:68004, February 2008.

[Ber92] S. K. Berberian. *Linear Algebra*. Oxford University Press, New York, 1992.

[BKC09] M. Boguna, D. Krioukov, and K. C. Claffy. Navigability of complex networks. *Nature Physics*, 5:74–80, 2009.

[BKO+02] S. Boccaletti, J. Kurths, G. Osipov, D. L. Valladares, and C. Zhou. The synchronization of chaotic systems. *Phys. Rep.*, 366:1, 2002.

[Bol01] B. Bollobas. *Random Graphs*. Cambridge University Press, 2 edition, 2001.

[Bol06] E. M. Bollt. Modeling spatiotemporal systems on arbitrary networks by uncovering spatial scales and component interactions: Uncovering hierarchical scale interactions for problems of mathematical epidemiology. Proposal to the US Army Research Office, 2006.

[Bol07] E. M. Bollt. Attractor modeling and empirical nonlinear model reduction of dissipative dynamical systems. *International Journal of Bifurcation and Chaos*, 17:1199, 2007.

[Bow75] R. Bowen. $\omega-$limit sets for axiom a diffeomorphisms. *J. Diff. Eqns.*, 18:333–339, 1975.

[BP02a] M. Barahona and L. M. Pecora. Synchronizaiton in small-world systems. *Phys. Rev. Lett.*, 89:054101, 2002.

[BP02b] M. Boguna and R. Pastor-Satorras. Epidemic spreading in correlated complex networks. *Phys. Rev. E*, 66:047104, 2002.

[BP03] M. Boguna and R. Pastor-Satorras. Class of correlated random networks with hidden variables. *Phys. Rev. E*, 68:036112, 2003.

[BZC+03] D. Bu, Y. Zhao, L. Cai, H. Xue, X. Zhu, H. Lu, J. Zhang, S. Sun, L. Ling, N. Zhang, G. Li, and R. Chen. Topological structure analysis of the protein-protein interaction network in budding yeast. *Nucleic Acids Res.*, 31:2443–2450, 2003.

[CCDM02] G. Caldarelli, A. Capocci, P. De Los Rios, and M. A. Munoz. Scale-free networks from varying vertex intrinsic fitness. *Phys. Rev. Lett.*, 89:258702, 2002.

[CCSb09] S. Carmi, S. Carter, J. Sun, and D. ben-Avraham. Asymptotic behavior of the kleinberg model. *Phys. Rev. Lett.*, 102:238702, June 2009.

[CD09] C. C. Cartozo and P. De Los Rios. Extended navigability of small world networks: Exact results and new insights. *Phys. Rev. Lett.*, 102:238703, 2009.

[CEbH00] R. Cohen, K. Erez, D. ben-Avraham, and S. Havlin. Resilience of the internet to random breakdowns. *Phys. Rev. Lett.*, 85:4626, 2000.

[CKY88] E. M. Coven, I. Kan, and J. A. Yorke. Pseudo-orbit shadowing in the family of tent maps. *Trans. Amer. Math. Soc.*, 308:227–241, 1988.

[CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2 edition, 2001.

[CLV03] F. Chung, L. Lu, and V. Vu. Spectra of random graphs with given expected degrees. *Proc. Natl. Acad. Sci. USA*, 100:6313, 2003.

[CS09] Y. Choi and W. Szpankowski. Compression of graphical structures. In *IEEE International Symposium on Information Theory (ISIT)*, pages 364–368, 2009.

[CT06] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2 edition, July 2006.

[DA05] J. Duch and A. Arenas. Community detection in complex networks using extremal optimization. *Phys. Rev. E*, 72:027104, 2005.

[Dem97] J. W. Demmel. *Applied Numerical Algebra*. SIAM, 1997.

[DGM08] S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Mendes. Critical phenoma in complex networks. *Rev. Mod. Phys.*, 80(4):1275–1335, October 2008.

[DHS00] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, 2 edition, October 2000.

[Die06] R. Diestel. *Graph Theory*. Springer, 3 edition, 2006.

[DMW03] P. S. Dodds, R. Muhamad, and D. J. Watts. An experimental study of search in global social networks. *Science*, 301(5634):827–829, 2003.

[EGSS82] S. C. Eisenstat, M. C. Gursky, M. H. Schultz, and A. H. Sherman. Yale sparse matrix package i: the symmetric codes. *Int. J. Nuer. Methods Eng.*, 18:1145, 1982.

[ER05] E. Estrada and J. A. Rodriguez-Velazquez. Subgraph centrality in complex networks. *Phys. Rev. E*, 71:056103, 2005.

[Fie89] M. Fiedler. Laplacian of graphs and algebraic connectivity. *Combinatorics and Graph Theory*, 25:57–70, 1989.

[Fis05] G. Fishman. *A First Course in Monte Carlo.* Duxbury Press, 1 edition, October 2005.

[FS91] J. D. Farmer and J. J. Sidorowich. Optimal shadowing and noise reduction. *Physica D*, 47:373–392, 1991.

[GBB09] A. Gautreau, A. Barrat, and M. Barthelemy. Microdynamics in stationary complex networks. *Proc. Natl. Acad. Sci. USA*, 106(22):8847–8852, June 2009.

[GD07] D. Gfeller and P. De Los Rios. Spectral coarse graining of complex networks. *Phys. Rev. Lett.*, 99:038701, 2007.

[GD08] D. Gfeller and P. De Los Rios. Spectral coarse graining and synchronization in oscillator networks. *Phys. Rev. Lett.*, 100:174104, 2008.

[GDD+03] R. Guimera, L. Danon, A. Diaz-Guilera, F. Giralt, and A. Arenas. Self-similar community structure in a network of human interactions. *Phys. Rev. E*, 68: 065103(R), 2003.

[GHYS90] C. Grebogi, S. H. Hammel, J. A. Yorke, and T. D. Sauer. Shadowing of physical trajectories in chaotic dynamics: containment and refinement. *Phys. Rev. Lett.*, 65:1527–1530, 1990.

[Gil98] I. Gilmour. Nonlinear model evaluation: l-shadowing, probabilistic prediction and weather forecasting. PhD thesis, Mathematical Institute, University of Oxford, 1998.

[Gol80] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, 1980.

[GSBC99] M. Golubitsky, I. Stewart, P-L. Buono, and J. J. Collins. Symmetry in locomoter central pattern generators and animal gaits. *Nature*, 401:693–695, October 1999.

[GV96] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Jonhs Hopkins University Press, 3 edition, 1996.

[Hay02] W. B. Hayes. Shadowing high-dimensional hamitonian systems: The gravitational n-body problem. *Phys. Rev. Lett.*, 90(5):054104, 2002.

[HIS81] P. L. Hammer, T. Ibaraki, and B. Simeone. Threshold sequences. *SIAM J. Alg. Disc. Meth.*, 2:39, 1981.

[HS08] A. Hagberg and D. A. Schult. Rewiring networks for synchronization. *Chaos*, 18:037105, 2008.

[HSS06] A. Hagberg, P. J. Swart, and D. A. Schult. Designing threshold networks with given structural and dynamical properties. *Phys. Rev. E*, 74:056116, 2006.

[HYG87] S. H. Hammel, J. A. Yorke, and C. Grebogi. Do numerical orbits of chaotic processes represent true orbits? *J. Complexity*, 3:136–145, 1987.

[HYG88] S. H. Hammel, J. A. Yorke, and C. Grebogi. Numerical orbits of chaotic processes represent true orbits. *Bull. Amer. Math. Soc.*, 19:465–470, 1988.

[Jos05] J. Jost. *Dynamical Systems: Examples of Complex Behavior*. Springer, 1 edition, September 2005.

[Jud08] K. Judd. Shadowing pseudo-orbits and gradient descent noise reduction. *Journal of Nonlinear Science*, 18(1):57–74, February 2008.

[Kim04] B. J. Kim. Geographical coarse graining of complex networks. *Phys. Rev. Lett.*, 93(16):168701, 2004.

[Kle99] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, September 1999.

[Kle00a] J. M. Kleinberg. Navigation in a small world. *Nature*, 406:845, 2000.

[Kle00b] J. M. Kleinberg. The small-world phenomenon: An algorithmic perspective. In F. Yao and E. Luks, editors, *Proceedings of the 32nd ACM Symposium on Theory of Computing*, pages 163–170. ACM, Portland, OR, 2000.

[KLK02] S-Y. Kim, W. Lim, and Y. Kim. Effect of parameter mismatch and noise on weak synchronization. *Progress of Theoretical Physics*, 107:239, 2002.

[KMRS05] N. Konno, N. Masuda, R. Roy, and A. Sarkar. Rigorous results on the threshold network model. *J. Phys. A: Math. Theor.*, 38:6277, 2005.

[KT05] J. M. Kleinberg and E. Tardos. *Algorithm Design.* Addison Wesley, 2005.

[Kur84] Y. Kuramoto. *Chemical oscillations, waves, and turbulence.* Springer-Verlag, New York, 1984.

[KW06] G. Kossinets and D. J. Watts. Empirical analysis of an evolving social network. *Science*, 311:88–90, 2006.

[LM05] A. N. Langville and C. D. Meyer. A survey of eigenvector methods for web information retrieval. *SIAM Review*, 47(1):135–161, 2005.

[LPC+07] E. Lopez, R. Parshani, R. Cohen, S. Carmi, and S. Havlin. Limited path percolation in complex networks. *Phys. Rev. Lett.*, 99:188701, 2007.

[LT85] P. Lancaster and M. Tismenetsky. *The Theory of Matrices with Applications.* Academic Press, 2 edition, 1985.

[Mac00] C. R. MacCluer. The many proofs and applications of perron's theorem. *SIAM Review*, 42(3):487–498, 2000.

[Mat] Mathworks. URL `http://www.mathworks.com`.

[Mer94] R. Merris. Laplacian matrices of graphs: a survey. *Linear Algebra Appl.*, 197&198:143–176, 1994.

[Mer03] R. Merris. Split graphs. *Eur. J. Comb.*, 24:413–430, 2003.

[MGK06] S. J. Moon, R. Ghanem, and I. G. Kevrekidis. Coarse graining the dynamics of coupled oscillators. *Phys. Rev. Lett.*, 96:144101, 2006.

[MH38] M. Morse and G. A. Hedlund. Symbolic dynamics. *American Journal of Mathematics*, 60:815–866, 1938.

[Mil67] S. Milgram. The small world problem. *Psychol. Today*, 2:60–67, 1967.

[MM07] P. N. McGraw and M. Menzinger. Analysis of nonlinear synchronization dynamics of oscillator networks by laplacian spectral methods. *Phys. Rev. E*, 75: 027104, 2007.

[MMK04] N. Masuda, H. Miwa, and N. Konno. Analysis of scale-free networks based on a threshold graph with intrinsic vertex weights. *Phys. Rev. E*, 70:036124, 2004.

[Moh91] B. Mohar. The laplacian spectrum of graphs. In *Graph Theory, Combinatorics, and Applications*, volume 2, pages 871–898. Wiley, 1991.

[MSN09] A. Milanese, J. Sun, and T. Nishikawa. Spectral impact of structural modifications in complex networks. (Manuscript in preparation), November 2009.

[Net] NetworkX. URL `https://networkx.lanl.gov/wiki`.

[New02] M. E. J. Newman. Assortative mixing in networks. *Phys. Rev. Lett.*, 89:208701, 2002.

[New03] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[New06] M. E. J. Newman. Modularity and community structure in networks. *Proc. Natl. Acad. Sci. USA*, 103:8577–8582, 2006.

[NM06a] T. Nishikawa and A. E. Motter. Maximum performance at minimum cost in network synchronization. *Physica D*, 224:77, 2006.

[NM06b] T. Nishikawa and A. E. Motter. Synchronization is optimal in non-diagonalizable networks. *Phys. Rev. E*, 73:065106(R), 2006.

[NY88] H. E. Nusse and J. A. Yorke. Is every approximate trajectory of some process near an exact trajectory of a neaby process? *Comm. Math. Phys.*, 114:363–379, 1988.

[Paj] Pajek. URL `http://vlado.fmf.uni-lj.si/pub/networks/pajek/`.

[Pal00] K. Palmer. *Shadowing in Dynamical Systems*. Kluwer Academic Publishers, 2000.

[PC90] L. M. Pecora and T. L. Carroll. Synchronization in chaotic systems. *Phys. Rev. Lett.*, 64:821, 1990.

[PC98] L. M. Pecora and T. L. Carroll. Master stability functions for synchronized coupled systems. *Phys. Rev. Lett.*, 80:2109, 1998.

[PCB06] S. D. Pethel, N. J. Corron, and E. M. Bollt. Symbolic dynamics of coupled map lattices. *Phys. Rev. Lett.*, 96:034105, 2006.

[Per96] L. Perko. *Differential Equations and Dynamical Systems*. Springer-Verlag, New York, 2 edition, 1996.

[Pil99] S. Yu. Pilyugin. *Shadowing in Dynamical Systems (Lecture Notes in Mathematics)*. Springer, October 1999.

[PKJ09] S. Pigolotti, S. Krishna, and M. H. Jensen. Symbolic dynamics of biological feedback networks. *Phys. Rev. Lett.*, 102:088701, 209.

[POM09] M. A. Porter, J.-P. Onnela, and P. J. Mucha. Communities in networks. *Notices of the AMS*, 56(9):1082–1097, 1164–1166, 2009.

[PRK01] A. Pikovsky, M. Rosenblum, and J. Kurths. *Synchronization: A universal concept in nonlinear sciences*. Cambridge University Press, Cambridge, 2001.

[PSBS06] M. Porfiri, D. J. Stilwell, E. M. Bollt, and J. D. Skufca. Random talk: Random walk and synchronizability in a moving neighborhood network. *Physica D*, 224 (1-2):102–113, 2006.

[PVV01] R. Pastor-Satorras, A. Vazquez, and A. Vespignani. Dynamical and correlation properties of the internet. *Phys. Rev. Lett.*, 87:258701, 2001.

[RCb02] A. F. Rozenfeld, R. Cohen, and D. ben-Avraham. Scale-free networks on lattices. *Phys. Rev. Lett.*, 89:218701, 2002.

[RJ02] D. Ridout and K. Judd. Convergence properties of gradient descent noise reduction. *Physica D*, 165:26–47, 2002.

[ROH04] J. G. Restrepo, E. Ott, and B. R. Hunt. Spatial patterns of desynchronization bursts in networks. *Phys. Rev. E*, 69:066215, 2004.

[ROH06a] J. G. Restrepo, E. Ott, and B. R. Hunt. Characterizing the dynamical importance of network nodes and links. *Phys. Rev. Lett.*, 97:094102, 2006.

[ROH06b] J. G. Restrepo, E. Ott, and B. R. Hunt. The emergence of coherence in complex networks of heterogeneous dynamical systems. *Phys. Rev. Lett.*, 96: 254103, 2006.

[ROH07] J. G. Restrepo, E. Ott, and B. R. Hunt. Approximating the largest eigenvalue of network adjacency matrices. *Phys. Rev. E*, 76:056119, 2007.

[Ros76] O. E. Rossler. An equation for continuous chaos. *Physics Letters A*, 57(5): 397–398, 1976.

[Rug96] W. J. Rugh. *Linear System Theory.* Prentice Hall, New Jeresy, 2 edition, 1996.

[Sau02] T. D. Sauer. Shadowing breakdown and large errors in dynamical simulations of physical systems. *Phys. Rev. E*, 65:036220, February 2002.

[SB04] J. D. Skufca and E. M. Bollt. Communication and synchronization in disconnected networks with dynamic topology – moving neighborhood networks. *Mathematical Biosciences and Engineering*, 1:2, 2004.

[Sb09] J. Sun and D. ben-Avraham. Greedy connectivity: A new notion of connectivity for space embedded networks. (Manuscript in preparation), December 2009.

[SBb08] J. Sun, E. M. Bollt, and D. ben-Avraham. Graph compression – save information by exploiting redundancy. *J. Stat. Mech.*, page P06001, June 2008.

[SBB+09] J. Sun, J. P. Bagrow, E. M. Bollt, J. D. Skufca, and E. Lopez. Efficient computation of path lengths for evolving networks. (Manuscript in preparation), November 2009.

[SBBS09] J. Sun, J. P. Bagrow, E. M. Bollt, and J. D. Skufca. Dynamic computation of network statistics via updating schema. *Phys. Rev. E*, 79:036116, March 2009.

[SBN09a] J. Sun, E. M. Bollt, and T. Nishikawa. Constructing generalized synchronization manifolds by manifold equation. *SIAM J. Appl. Dyn. Syst.*, 8(1):202–221, January 2009.

[SBN09b] J. Sun, E. M. Bollt, and T. Nishikawa. Judging quality of model reduction via shadowing criteria. (Manuscript in preparation), November 2009.

[SBN09c] J. Sun, E. M. Bollt, and T. Nishikawa. Master stability functions for coupled nearly identical dynamical systems. *Euro. Phys. Lett.*, 85:60011, April 2009.

[SBN09d] J. Sun, E. M. Bollt, and T. Nishikawa. Synchronization stability of coupled near-identical oscillator network. In Jie Zhou, editor, *Complex 2009, Part I, LNICST*, volume 4, pages 900–911. Springer Berlin Heidelberg, June 2009.

[SBPD09] J. Sun, E. M. Bollt, M. A. Porter, and M. S. Dawkins. Synchronization of cows. (Manuscript in preparation), December 2009.

[SBR06] D. J. Stilwell, E. M. Bollt, and D. G. Roberson. Sufficient conditions for fast switching synchronization in time-varying network topologies. *SIAM J. Appl. Dyn. Syst.*, 5(1):140–156, March 2006.

[SGL07] J. Sun, Y. Ge, and S. Li. Evolving network with different edges. *Phys. Rev. E*, 76:046108, October 2007.

[SGP03] I. Stewart, M. Golubitsky, and M. Pivato. Symmetry groupoids and patterns of synchrony in coupled cell networks. *SIAM J. Appl. Dyn. Syst.*, 2(4):609–646, 2003.

[SGY97] T. D. Sauer, C. Grebogi, and J. A. Yorke. How long do numerical chaotic solutions remain valid? *Phys. Rev. Lett.*, 79(1):59–62, 1997.

[SNb08] J. Sun, T. Nishikawa, and D. ben-Avraham. Sequence nets. *Phys. Rev. E*, 78: 026104, August 2008.

[SS93] S. H. Strogatz and I. Stewart. Coupled oscillators and biological synchronization. *Scientific American*, 269:102, 1993.

[Str00] S. H. Strogatz. From kuramoto to crawford: exploring the onset of synchronization in populations of coupled oscillators. *Physica D*, 143:1–20, 2000.

[Tsa] P. Tsaparas. URL `http://www.cs.helsinki.fi/u/tsaparas/MACN2006/1\verb1data-code.html1`.

[TSL00] J. B. Tenenbaum, Vin de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319, 2000.

[TY71] R. Tarjan and A. Yao. Storing a sparse table. *Commun. ACM*, 22:606, 1971.

[VHO96a] S. C. Venkataramani, B. R. Hunt, and E. Ott. Bubbling transition. *Phys. Rev. E*, 54:1346, 1996.

[VHO+96b] S. C. Venkataramani, B. R. Hunt, E. Ott, D. J. Gauthier, and J. C. Biefang. Transitions to bubbling of chaotic systems. *Phys. Rev. Lett.*, 77:5361, 1996.

[VPV02] A. Vazquez, R. Pastor-Satorras, and A. Vespignani. Large-scale topological and dynamical properties of the internet. *Phys. Rev. E*, 65:066130, 2002.

[Wat99] D. J. Watts. *Small Worlds: The Dynamics of Networks between Order and Randomness*. Princeton University Press, August 1999.

[Wes00] D. B. West. *Introduction to Graph Theory*. Prentice Hall, 2 edition, September 2000.

[Wil65] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Oxford University Press, Oxford, UK, 1965.

[WS98] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440, 1998.

[WSSV85] A. Wolf, J. B. Swift, H. L. Swinney, and J. A. Vastano. Determing lyapunov exponents from a time series. *Physica D*, 16:285–317, 1985.

[YB07] C. Yao and E. M. Bollt. Modeling and nonlinear parameter estimation with kronecker product representation for coupled oscillators and spatiotemporal systems. *Physica D*, 227:78–99, 2007.

[Zho06] S. Zhou. Understanding the evolution dynamics of internet topology. *Phys. Rev. E*, 74:016124, 2006.