Estimation on a Partially Observed Influence Model

by

Carlos Alberto Gómez Uribe

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

June 2003

Author .....
Department of Electrical Engineering and Computer Science
June 6, 2003

Certified by .....
George C. Verghese
Professor
Thesis Supervisor

Accepted by ..
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Estimation on a Partially Observed Influence Model

by

Carlos Alberto Gómez Uribe

Submitted to the Department of Electrical Engineering and Computer Science
on June 6, 2003, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

The influence model is a mathematical representation of stochastic interactions on networks. It consists of a directed graph where each node (or site) is like a Markov chain with transition probabilities that depend on its present state and that of its neighbors. This thesis introduces the partially observed influence model (POIM), an influence model where the state of only some nodes in the graph is known at each time. The state of these observed sites at each time is referred to as an observation. Motivated by the success of HMMs as modeling tools in science and engineering, this thesis aims to develop efficient methods to address the following problems:

1. Find the probability of an observation sequence.
2. Given a sequence of observations, finding the most likely state of the network.
3. Estimating the present state of the network given the observations up to the present.
4. Estimating the parameters of a POIM given a sequence of observations.

Efficient methods that solve the first three problems are developed using two main approaches. The first one uses methods that have been successful in similar problems related to the HMM. The second one analyzes and takes advantage of the graph structure of the network.

A method to approach the fourth problem for a particular class of POIMs, the homogeneous POIMs, is also presented. This method, although not always accurate, can be useful to find reasonable parameters of a POIM based on a sequence of observations.

We hope that the methods developed here will contribute to make the influence model a useful model of networked interactions.

Thesis Supervisor: George C. Verghese
Title: Professor
Associate Supervisor : Sandip Roy
Title: PhD

Dedicada a:

mi chinita linda,

papá, mamá,

y al apestoso Chingardo.

Acknowledgments

First of all, I want to thank Prof. George Verghese. He has been a wonderful thesis advisor, always pointing me in the right direction, sharing his wisdom to help me find the right questions, encouraging me to go on listening to my ideas and dazzling me with his, and making engineering fun. More importantly, he has also been a great friend: patient, helpful, encouraging and understanding. Thanks George!

I am very grateful to Sandip Roy, who is the associate supervisor of this thesis. He gave invaluable comments and contributed greatly to the content, most of which was conceived during the weekly meetings George, Sandip and I had. Sandip suggested many successful ideas of further research, and motivated me to think about these problems in fruitful ways. His help proofreading and revising the thesis is greatly appreciated.

Many thanks to Prof. John Bush for showing me how much fun science can be, and for teaching me fluid dynamics. I am also grateful to Prof. Alar Toomre for his funny and colorful lectures on differential equations and complex variables. Many thanks to my academic advisor, Prof. Rafael Reif, for always letting me add and drop what I wanted. I am also thankful to the DoD for sponsoring this research under an initiative to study complex systems.

También le doy las gracias a todos mis amigos en MIT, y en México por su amistad y por todos los momentos que compartimos.

Quiero darle las gracias a mi familia por haberme forjado el carácter, por haberme brindado las oportunidades de hacer lo que siempre quise, y por siempre decirme las cosas como son. Le agradezco a mi papá por enseñarme a disfrutar de lo cotidiano, a poner las cosas en perspectiva y por siempre mantenerse en contacto. Le agradezco a mi mamà la devoción maternal y el cariño con el que me crió, por contagiarme su sed de conocimiento, y por enseñarme, a base de una incesante repetición, lo que son los verbos irregulares por diptongación. Gracias al fabuloso pero apestoso Chinguer por ser el mejor hermano que pude haber

tenido, aunque te tardaste en llegar huesudo. Me enorgullece ser tu hermano y realmente espero que en el futuro podamos pasar más tiempo juntos.

Finalmente quiero agradecerle a Rebex, mi chinita preciosa (y ahora mi comprometida), por darme la oportunidad de disfrutar de lo cotidiano, por llenar cada uno de mis amaneceres de felicidad, y por siempre estar ahí conmigo, apoyándome, y queriéndome tanto e incondicionalmente (aunque quiera ser profesor). Gracias por darme el sueño de una vida feliz a tu lado. Te amo. Ah, sí, gracias por llevarme a Stanford y por aguantarme y por todas las demás cosas bonitas que haces por mí.

Contents

List of Figures

Chapter 1

Introduction and Overview

1.1 Introduction

Stochastic models have become useful and common tools in science and engineering. An example is the Hidden Markov Model (HMM), which consists of a Markov chain whose state cannot be observed directly, but only through the emission of symbols with state dependent probabilities. HMMs have found widespread use in a range of disciplines, such as in speech recognition, mostly because of the existence of techniques that make them easy to work with. The existence of efficient techniques to estimate the state and parameters of an HMM by measuring a sequence of observations accounts for much of the popularity of HMMs.

Recently, an effort has been made to use stochastic models to understand *networked* systems. In such systems, however, the number of possible states that the system may take at a given time often becomes too large, which makes general Markov models very hard to work with. This difficulty has promoted the creation of new stochastic models that are better suited for networked systems. The Influence Model (IM) is one such example. The main goal of this thesis will be to develop estimation techniques for partially observed IMs; analogous to those that render HMMs useful.

HMMs are models that have an underlying Markov chain. As this Markov chain transitions from one time step to the next, it emits a symbol whose probabilistic description is solely

a function of the present state of the chain. Therefore in an HMM there is an underlying stochastic process (the Markov chain) that is hidden, and that can only be observed through another stochastic process that produces the sequence of observations. A good introduction to these models is provided in [9], and an overview of related literature can be found in [6]. As discussed in [9], there exist efficient techniques that are used in HMMs to find answers to the three following questions:

1. What is the probability that a particular sequence of observations occurs?

2. How do we efficiently estimate the state of the HMM, given a sequence of observations?

3. How do we infer the transition and emission probabilities that best explain a sequence of observations?

The fact that there exist efficient techniques to answer these three questions make HMMs useful in real-world applications.

The IM was introduced and is extensively discussed in [1]. (For a concise introduction to the IM, refer to [2].) It consists of a directed graph where each node (or vertex or site) is a Markov-like chain. Each node is in one of a finite number of statuses at each time, and status occupancies at a node at the next time instant are governed by a probability mass function (pmf). However, while in a Markov chain the pmf is determined by its present status, in an IM the pmf is also affected by the present status of each of its neighbors. More specifically, each node recieves at each time one status-dependent pmf from each of its neighbors and itself, and uses a convex combination of these as the pmf that governs its status occupancies at the next time instant. Therefore, the pmf of each node in the network at any given time is a function of the status of that node, and that of its neighbors. The fact that this pmf is a linear combination of other pmfs makes the model particularly tractable but limits its modeling power, since many applications have nonlinear behavior. Although to our knowledge the IM is not widely used yet, it has been tried in a couple of scenarios. We believe the IM has potential as a modeling tool for some particular networked systems. It has been used to model propagation of failures in a power grid [2], and to model the

interactions of individuals in a room (see [4]), and to measure the impact of online (i.e., through the Internet and such) social interactions on the use of digital libraries (see [10]).

With the hope of making the IM appealing to model some of these problems we introduce the notion of a Partially Observed Influence Model (POIM). The POIM is an influence model where only the state of a subset of the nodes in the graph is known at each time. We thus have a system that outputs a sequence of observations but with an unknown overall state. It is in this context that we pose the following three questions, analogous to those posed for HMMs:

1. What is the probability that a particular sequence of observations occurs?

2. How do we efficiently estimate the state of the network given a sequence of observations?

3. How do we infer the probabilities (or parameters) of the IM that best explain a sequence of observations?

The first question can arise in the following context: Imagine there is a system that you know is one of two IMs and you have followed it for some time and thus have some sequences of observations. You may wish to decide which of the two possible IMs is more likely to have created the sequence of observations, and for that you need to answer the first question. The second question tries to estimate the state of the network based on the partial information you have about it. And the third question is one of system identification. We believe that a successful development of efficient techniques that address these questions will make the IM more appealing as a real modeling tool for networked systems. That is the purpose of this thesis.

In this thesis we answer these three questions, adapting the techniques developed for HMMs and taking advantage of the graph structure of the IM whenever possible. An answer to the third question was found only for a particular kind of IMs called homogeneous.

1.2 Overview

In Chapter 2 we present a review the techniques used for HMMs, as they provide a basis for the main problems we wish to address. We also re-introduce the IM, introduce the POIM, and formulate the three problems we wish to address in the rest of the thesis. In Chapter 3 we present a solution to the three problems of interest, based on the techniques used for HMMs. These solutions do not take advantage of the graph structure of the IM. In Chapter 4 we present ways to take advantage of the network structure in a POIM to solve the first two problems in a much more efficient way than in Chapter 3. We also develop efficient algorithms to do online estimation of the present state of the network (or of a set of sites in the graph) given the observations up to that time. Finally Chapter 5 summarizes the main contributions of this thesis and suggests some directions for future work. Specific examples are carried along in the thesis to illustrate the use of the techniques presented.

Chapter 2

Background and Problem Formulation

2.1 Review of Hidden Markov Models (HMMs)

An HHM involves an underlying and unobserved sequence of states governed by Markov chain with finite state space, and an accompanying sequence of observations whose probability distribution at any time is determined only by the current state of that Markov chain. HMMs have been used successfully in a wide variety of applications, such as in speech recognition ([9]), and as models of homicides in Cape Town, South Africa ([8]). Recently, HMMs have been very successful in Biology where they have been used, for example, to identify families of proteins ([5]), to classify proteins according to these families ([3]), to find protein-coding genes ([3]) and determine gene structure ([7]), and to do multiple sequene alignments ([7]).

In part, the success of HMMs is due to the fact that the likelihood of even a long sequence of observations can be computed efficiently enough to enable parameters to be estimated by direct numerical maximization of that likelihood. Here, we review the basic theory behind HMMs because it will be relevant to the problems we address. In particular, we build on the same algorithms that are used for HMMs (the Forward-Backward, the Viterbi, and the Baum-Welch algorithms) so we introduce them here. A reader who is familiar with these

algorithms may skip to the next section. In this section, we first characterize an HMM and then proceed to present what Rabiner [9] calls the three main problems of HMMs, with their respective solutions. The material in this section follows closely that presented in [9].

2.1.1 HMM Definition

For the purposes of this thesis, we specify an HMM by the following elements:

- The number of states that the underlying Markov chain can take, n. These states cannot be observed directly, but through emitted symbols. However, these states usually have some physical significance. We denote the individual states that the chain can take as $C = c_1, c_2, ..., c_n$, and the actual state at time t as Q^t. We denote a sequence of states as $Q^{l,k} = Q_l Q_{l+1} ... Q_k$, and so a sequence of states from time 1 to time t is written as $Q^{1,t}$.

- The number of distinct observations per state, m. These observation symbols are the physical output of the system that is being modeled. The individual symbols will be denoted by $V = v_1, v_2, ..., v_m$, and the symbol emitted at time t as O^t. Throughout this thesis, we will denote a sequence of observations as $O^{l,k} = O^l O^{l+1} ... O^k$, so a sequence of observations from time 1 to time t is written as $O^{1,t}$.

- The state transition probabilities $\mathbf{A} = \{a_{ij}\}$ where

$$a_{ij} = P\big(Q^{t+1} = c_j | Q^t = c_i\big), \qquad 1 \leq i, j \leq n. \tag{2.1.2.1}$$

- The probability distribution of the observation symbols $\mathbf{B} = \{b_i(v_k)\}$, where

$$b_i(v_k) = P\big(O^t = v_k | Q^t = c_i\big), \quad 1 \leq j \leq n; \quad 1 \leq k \leq m. \tag{2.1.2.2}$$

- The initial state distribution $\pi = \{\pi_i\}$ where

$$\pi_i = P(Q^1 = c_i), \qquad 1 \leq i \leq n. \tag{2.1.2.3}$$

Therefore, a complete specification of an HMM requires specification of two model parameters (m and n), specification of the alphabet of observed symbols, and specification of three probability measures ($\mathbf{A}, \mathbf{B}$, and π). We use the notation $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ to denote the complete parameter set of the model.

Once an HMM has been completely specified, it can be used to generate a sequence of observations over the interval $1, T$ in the following way:

1. An initial state $Q^1 = c_i$ is chosen according to the initial distribution π.

2. Set $t = 1$.

3. Choose $O^t = v_k$ according to the symbol probability in state c_i, i.e., b_i.

4. Transit to a new state $Q^{t+1} = c_j$ according to the state transition probability for state c_i, i.e., a_{ij}.

5. Set $t = t + 1$; return to step 3) if $t < T$; otherwise terminate the procedure.

2.1.2 The Three Basic Problems for HMMs

According to Rabiner[9], there are three main problems of interest that had to be solved for HMMs to be useful in real applications. These are the following:

1. Given an observation sequence $O^{1,T}$, and an HMM λ, how do we efficiently calculate $P(O^{1,T}|\lambda)$, the likelihood of the observation sequence, given the model?

2. Given an observation sequence $O^{1,T}$, and the HMM λ, how do we choose the state sequence $Q^{1,T} = Q_1 Q_2 ... Q_T$ that *best* explains the observations? Or how do we find the most likely state Q^k given a sequence of observations?

3. How do we adjust the model parameters $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ to maximize $P(O^{1,T}|\lambda)$?

Problem 1 evaluates how well a given model λ matches a given set of observations. If there are different candidate HMMs for a particular set of observations, we can solve for the likelihood of the observations given each of the models, and then choose the candidate for which the likelihood is the greatest.

Problem 2 tries to uncover the hidden states of the model. Different criteria can be used to find the sequence of states that *best* explains an observation sequence. There are two that are very common. The first is called *local decoding* of the state at time t, and refers to the determination of the state Q^t that is maximum *a posteriori* (MAP):

$$Q^t = \arg \max_{1 \leq i \leq n} P(Q^t = c_i | O^{1,T}).$$

The other approach is referred to as *global* MAP *decoding* and it determines the *sequence* of states $Q^{1,T}$ that maximizes the conditional probability $P(Q^{1,T}|O^{1,T})$, or more conveniently, the joint probability $P(Q^{1,T}, O^{1,T})$.

Problem 3 is the hardest one. It tries to find the model that can best account for the output observations. It tries to uncover the parameters of the model, given the observations.

There are four properties of the HMM that are used repeatedly in the solution of these three problems, and they are stated below:

1. Firstly, for $1 \leq t \leq T$,

$$P(O^{1,T}|Q^t) = P(O^{1,t}|Q^t)P(O^{t+1,T}|Q^t), \tag{2.1.2.4}$$

   where we use the convention that when $t = T$ then $P(O^{t+1,T}|Q^t) = 1$.

2. Secondly, for $1 \leq t \leq T-1$,

$$P(O^{1,T}|Q^t, Q^{t+1}) = P(O^{1,t}|Q^t)P(O^{t+1,T}|Q^{t+1}). \tag{2.1.2.5}$$

3. Thirdly, for $1 \leq t \leq l \leq T$; we have that

$$P(O^{l,T}|Q^{t,l}) = P(O^{l,T}|Q^{l}). \tag{2.1.2.6}$$

4. Finally, for $1 \leq t \leq T$ we have that

$$P(O^{t,T}|Q^{t}) = P(O^{t}|Q^{t})P(O^{t+1,T}|Q^{t}). \tag{2.1.2.7}$$

With these four properties in mind, we can proceed to explain the solution of the three main problems for HMMs.

2.1.3 The Forward-Backward Algorithm

We wish to solve for the probability of an observation $O^{1,T}$ given an HMM λ. The most straightforward way to do this would be to enumerate all the possible state sequences $Q^{1,T}$, calculate the joint probability of the observation and of each state sequence, and finally add up the joint probabilities for all the possible state sequences. This approach, however, is very slow since it involves on the order of Tn^T calculations (there are n^T possible state sequences, and it takes $2T$ calculations to evaluate the probability for each one). For a more detailed explanation of this approach, please refer to [9].

This problem is solved in a much more efficient way by using the Forward-backward Algorithm. This essentially uses dynamic programming to compute the *forward probabilities* $\alpha_t(i)$ and *backward probabilities* $\beta_t(i)$, so called because they require a forward and a backward pass through the data, respectively. These probabilities, given the model λ, are defined as

$$\alpha_t(i) = P(O^{1,t}, Q^t = c_i),$$

and

$$\beta_t(i) = P(O^{t+1,T}|Q^t = c_i).$$

The forward probability is thus the probability of the partial observation $O^{1,t} = O^1...O^t$, and of state c_i at time t, given the model λ. The backward probability is the probability of the partial observation sequence from $t+1$ to the end, given that the state is c_i at time t. Notice that because of the convention mentiones below (2.1.2.4), we have that $\beta_T(i) = 1$ for all i.

From these definitions, one may solve for the probability of an observation sequence by noting that for $1 \leq t \leq T$:

$$\begin{aligned} \alpha_t(i)\beta_t(i) &= P(O^{1,t}, Q^t = c_i)P(O^{t+1,T}|Q^t = c_i) \\ &= P(Q^t = c_i)P(O^{1,t}|Q^t = c_i)P(O^{t+1,T}|Q^t = c_i) \\ &= P(Q^t = c_i)P(O^{1,T}|Q^t = c_i), \qquad \text{by (2.1.2.4)} \\ &= P(O^{1,T}, Q^t = c_i), \end{aligned} \tag{2.1.2.8}$$

and

$$\sum_{i=1}^{n} \alpha_t(i)\beta_t(i) = P(O^{1,T}). \tag{2.1.2.9}$$

Therefore if we can evaluate the forward and the backward probabilities we have T ways to calculate the probability of an observation sequence, given λ. For the case when $t = T$, Equation (2.1.2.9) becomes $P(O^{1,T}) = \sum_{i=1}^{n} \alpha_T(i)$.

We use the Forward-Backward algorithm to find all the forward and the backward probabilities. To calculate the forward variables we do the following:

1. Initialization:

$$\alpha_1(i) = \pi_i b_i(O^1), \qquad 1 \leq i \leq n. \tag{2.1.2.10}$$

2. Induction:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{n} \alpha_t(i) a_{ij}\right] b_j(O^{t+1}), \quad 1 \leq t \leq T-1; \quad 1 \leq j \leq n. \tag{2.1.2.11}$$

Similarly, to calculate the backward probabilities the following algorithm is used:

1. Initialization:

$$\beta_T(i) = 1, \quad 1 \leq i \leq n. \tag{2.1.2.12}$$

2. Induction:

$$\beta_t(i) = \sum_{j=1}^{n} a_{ij} b_j(O^{t+1}) \beta_{t+1}(j), \quad T-1 \geq t \geq 1; \quad 1 \leq i \leq n. \tag{2.1.2.13}$$

Each of these algorithms takes on the order of n^2T calculations to compute all the forward and the backward variables, respectively. So we can compute the probability of an observation sequence $O^{1,T}$ in only order n^2T time, as opposed to the $2Tn^T$ of the first approach that was mentioned.

The remainder of this section shows why the recursions in the algorithms above hold, and may be skipped. For the forward variables we have that

$$
\begin{aligned}
\alpha_{t+1}(j) &= \sum_{i=1}^{n} P(O^{1,t+1},\ Q^t = c_i,\ Q^{t+1} = c_j) \\
&= \sum_{i=1}^{n} P(Q^t = c_i,\ Q^{t+1} = c_j) P(O^{1,t+1} | Q^t = c_i,\ Q^{t+1} = c_j) \\
&= \sum_{i=1}^{n} P(Q^t = c_i) a_{ij} P(O^{1,t} | Q^t = c_i) P(O^{t+1} | Q^{t+1} = c_j), \qquad \text{by (2.1.2.5)} \\
&= \sum_{i=1}^{n} P(O^{1,t}, Q^t = c_i) a_{ij} b_j(O^{t+1}) \\
&= \left(\sum_{i=1}^{n} \alpha_t(i) a_{ij} \right) b_j(O^{t+1}).
\end{aligned}
$$

Similarly, for the backward variables the induction relation is obtained as follows:

$$
\begin{aligned}
\beta_t(i) &= P(O^{t+1,T} | Q^t = c_i) \\
&= P(O^{t+1,T},\ Q^t = c_i) / P(Q^t = c_i) \\
&= \sum_{j=1}^{n} P(O^{t+1,T},\ Q^t = c_i,\ Q^{t+1} = c_j) / P(Q^t = c_i) \\
&= \sum_{j=1}^{n} P(O^{t+1,T} | Q^t = c_i,\ Q^{t+1} = c_j) \times P(Q^t = c_i,\ Q^{t+1} = c_j) / P(Q^t = c_i) \\
&= \sum_{j=1}^{n} P(O^{t+1,T} | Q^{t+1} = c_j) a_{ij}, \qquad \text{by (2.1.2.6), with } l = t+1. \\
&= \sum_{j=1}^{n} P(O^{t+1} | Q^{t+1} = c_j) P(O^{t+2,T} | Q^{t+1}) a_{ij}, \qquad \text{by (2.1.2.7).} \\
&= \sum_{j=1}^{n} b_j(O^{t+1}) \beta_{t+1}(j) a_{ij}.
\end{aligned}
$$

The Forward-Backward algorithm constitutes the most efficient solution to this problem that is known.

2.1.4 Local Decoding

In Problem 2 we wish to find the state sequence $Q^{1,T}$ that *best* explains an observation sequence $O^{1,T}$, given a model λ. Different approaches may be followed, depending on what is meant by the previous sentence. As mentioned before one approach is to choose, for each time index t, the state Q^t that is individually most likely. This maximizes the expected number of correct individual states in the sequence $Q^{1,T}$, and is called *local decoding.* One can do this simply by using the forward and backward variables from Problem 1 as follows: The variable $\gamma_t(i)$ is defined as

$$\gamma_t(i) = P(Q^t = c_i | O^{1,T}),$$

which is the probability of being in state c_i at time t, given the observations and the model. In terms of the forward and backward variables, we have:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O^{1,T})} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{n} \alpha_t(i)\beta_t(i)},$$

where the $\sum_{i=1}^{n} \alpha_t(i)\beta_t(i)$ in the denominator normalizes $\gamma_t(i)$ so that it is a probability measure (i.e., $\sum_{i=1}^{n} \gamma_t(i) = 1$). Now using $\gamma_t(i)$ we solve for the individually most likely state at time t, as follows:

$$Q^t = \arg \max_{1 \le i \le n} \left[\gamma_t(i)\right], \qquad 1 \le t \le T.$$

Using this method, it would take on the order of nT extra calculations (after the forward and backward variables have been found) to find all the individual states that are most likely. However, this solution may not be what is desired, because the sequence of states that this method produces can potentially be an invalid one for the model. For example, this method may tell us that the most likely state at time t is c_1 and at time $t+1$ is c_4, even when such a transition might be impossible according to the model (i.e., when $a_{14} = 0$). A second approach to this problem is therefore finding the single most likely *sequence* of states, i.e., to maximize $P(Q^{1,T}|O^{1,T})$, which is the same as maximizing $P(Q^{1,T}, O^{1,T})$. This approach is referred to as *global decoding,* and is described next.

2.1.5 The Viterbi Algorithm (Global Decoding)

The algorithm that is used to solve this problem is called the Viterbi algorithm, and uses dynamic programming to find the single best sequence of states $Q^{1,T}$ for the observation sequence $O^{1,T}$. We begin by defining the quantity

$$\delta_t(i) = \max_{Q^{1,t-1}} P\big(Q^{1,t-1}, Q^t = c_i, O^{1,t}\big),$$

which is the highest probability for a sequence of states from time 1 to $t-1$ that accounts for the first t observations, given the model, and whose final state is $Q^t = c_i$. Then, by induction we have that

$$\delta_{t+1}(j) = \big[\max_i \delta_t(i)a_{ij}\big] b_j(O^{t+1}).$$

Finally, to retrieve the actual path that yielded the highest probability for each t and for each j, we need to keep track of the previous state in each path via the array $\psi_t(j)$.

The complete procedure of the Viterbi algorithm is stated as follows in [9]:

1. Initialization:

$$\delta_1(i) = \pi_i b_i(O^1), \qquad 1 \le i \le n \tag{2.1.2.14}$$

$$\psi_1(i) = 0. \tag{2.1.2.15}$$

2. Induction:

$$\delta_t(j) = \max_{1\le i\le n} \big[\delta_{t-1}(i)a_{ij}\big] b_j(O^t), \qquad 2 \le t \le T, \quad 1 \le j \le n \tag{2.1.2.16}$$

$$\psi_t(j) = \arg\max_{1\le i\le n} \big[\delta_{t-1}(i)a_{ij}\big], \qquad 2 \le t \le T, \quad 1 \le j \le n. \tag{2.1.2.17}$$

3. Termination:

$$\begin{aligned} P^* &= \max_{1\leq i\leq n} \left[\delta_T(i)\right] \\ Q^{T*} &= \arg\max_{1\leq i\leq n} \left[\delta_T(i)\right]. \end{aligned} \tag{2.1.2.18}$$

4. Path (state sequence) backtracking:

$$Q^{t*} = \psi_{t+1}(Q^{t+1*}), \quad T-1 \geq t \geq 1. \tag{2.1.2.19}$$

The Viterbi algorithm is very similar to the Forward-Backward in that it uses dynamic programming. It also takes on the order of n^2T calculations.

2.1.6 The Baum-Welch Algorithm

The third problem is to determine a method to adjust the model parameters $\lambda = (A, B, \pi)$ to maximize the probability of a sequence of observations given the model. According to Rabiner ([9]), this problem is by far the most difficult of the three. He argues that, given a finite observation sequence, there is in fact no good way to determine the optimal model parameters. What can be done, however, is to first choose some arbitrary parameters $\lambda = (A, B, \pi)$ and tune them until $P(O^{0,T}|\lambda)$ is locally maximized. This is achieved via the Baum-Welch iteration method, or via gradient techniques. Unfortunately, in most problems of interest, the optimization surface is very complex and has many local maxima.

Here, we describe the Baum-Welch procedure to tune the HMM parameters, as is presented in [9]. First, we define $\xi_t(i,j)$ as the probability of being in state c_i at time t and state c_j at time $t+1$, given the model and the observation sequence, i.e.,

$$\xi_t(i,j) = P(Q^t = c_i, Q^{t+1} = c_j | O^{1,T}, \lambda).$$

Notice that in the previous sections we wrote probabilities given the model λ without writing explicitly that the model parameters were given. In this section we will make the notation explicit. This is because here we will have equations for probabilities relating two different

HMMs with different parameters, and we wish to make clear which one is used for each quantity.

From the definitions of the forward and the backward variables, we can write $\xi_t(i,j)$ as:

$$\begin{aligned}\xi_t(i,j) &= \frac{P(Q^t = c_i, Q^{t+1} = c_j, O^{1,T}|\lambda)}{P(O^{1,T}|\lambda)} \\ &= \frac{\alpha_t(i)a_{ij}b_j(O^{t+1})\beta_{t+1}(j)}{P(O^{1,T}|\lambda)} \\ &= \frac{\alpha_t(i)a_{ij}b_j(O^{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_t(i)a_{ij}b_j(O^{t+1})\beta_{t+1}(j)}. \end{aligned} \tag{2.1.2.20}$$

We had previously defined $\gamma_t(i)$ as the probability of being in state c_i at time t given the observation sequence and the model; so we can relate this variable to $\xi_t(i,j)$ by

$$\gamma_t(i) = \sum_{j=1}^{n} \xi_t(i,j). \tag{2.1.2.21}$$

If we sum $\gamma_t(i)$ over t we get a quantity that can be interpreted as the expected number of times that state c_i was visited, or equivalently, the number of transitions from state c_i if we exclude the time slot $t = T$ from the summation. In a similar way, summing $\xi_t(i,j)$ over t can be interpreted as the expected number of transitions from state c_i to state c_j. Thus we may write

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{expected number of transitions from state } c_i \tag{2.1.2.22}$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{expected number of transitions from } c_i \text{ to } c_j. \tag{2.1.2.23}$$

Using the above formulas a set of reasonable estimates for $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ are:

$$\begin{aligned}
\overline{\pi}_i &= \text{expected frequency in state } c_i \text{ at } t = 1 \\
&= \gamma_1(i) && (2.1.2.24) \\
\overline{a}_{ij} &= \frac{\text{expected number of transitions from state } c_i \text{ to state } c_j}{\text{expected number of transitions from state } c_i} \\
&= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} && (2.1.2.25) \\
\overline{b}_j(v_k) &= \frac{\text{expected number of times in state } c_j \text{ and observing symbol } v_k}{\text{expected number of times in state } c_j} \\
&= \frac{\sum_{t=1,\text{s.t. } O^t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}. && (2.1.2.26)
\end{aligned}$$

It has been shown that if one uses $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ to calculate the estimated parameters $\overline{\lambda} = (\overline{\mathbf{A}}, \overline{\mathbf{B}}, \overline{\pi})$ as defined by the above equations, then either the initial model λ defines a critical point of the likelihood, in which case $\lambda = \overline{\lambda}$; or the model $\overline{\lambda}$ is more likely than model λ in the sense that $P(O^{1,T}|\overline{\lambda}) > P(O^{1,T}|\lambda)$. So using this procedure, the Baum-Welch algorithm, we have found a new model $\overline{\lambda}$ that can (locally) only improve the likelihood of the observation sequence.

Based on the above procedure, one may recursively use $\overline{\lambda}$ instead of λ and continually improve the likelihood of $O^{1,T}$ being produced by the model, until some limiting point is reached. The final result of this procedure may be called a maximum likelihood local estimate of the HMM, because it is just a local maximum. Depending on how the initial parameters are chosen, different local maxima may be reached. Equations (2.1.2.24), (2.1.2.25), and (2.1.2.26) may also be derived by gradient techniques (see [9]). This iterative method to adjust the parameters is the best known solution to the problem, although it is not completely satisfying.

2.2 The Influence Model (IM)

The IM was introduced and is extensively discussed in [1], and will provide the context for this thesis. The IM is comprised of a network of n interacting nodes (or vertices, or sites). Site number i assumes one of a finite number m_i of possible statuses at each discrete-time step. The status of site i at time k is represented by a length-m_i column vector. This vector contains a single 1 in the position corresponding to its present status and a 0 everywhere else:

$$\mathbf{s}_i'[k] = [0 \ldots 010 \ldots 0].$$

The status of each of the n sites is updated at each time step, in a process that can be thought of as having the following three stages ([2]):

1. Each site i randomly selects one of its neighboring sites in the network–or itself– to be its determining site for stage 2; site j is selected as the determining site for site i with probability d_{ij} (so $d_{ij} \geq 0$ and $\sum_j d_{ij} = 1$).

2. Site j fixes the probability vector $\mathbf{p}_i'[k+1]$ for site i, depending on its status $\mathbf{s}_j'[k]$. The probability vector $\mathbf{p}_i'[k+1]$ will be used in the next stage to choose the next status of site i, so $\mathbf{p}_i'[k+1]$ is a length-m_i row vector with nonnegative entries that sum to 1. Specifically, if site j is selected as the determining site, then $\mathbf{p}_i'[k+1] = \mathbf{s}_j'[k]\mathbf{A}_{ji}$, where $\mathbf{A}_{ji}$ is a fixed row-stochastic $m_j \times m_i$ matrix. That is, $\mathbf{A}_{ji}$ is a matrix whose rows are probability vectors, with nonnegative entries that sum to 1.

3. The next status $\mathbf{s}_i'[k+1]$ is chosen according to the probabilities in $\mathbf{p}_i'[k+1]$, which was computed in the previous stage.

From the previous description it may be seen that the next-status probability vector for site i, given the status of its neighbors, is found by

$$\mathbf{p}_i'[k+1] = d_{i1}\mathbf{s}_1'[k]\mathbf{A}_{1i} + \ldots + d_{in}\mathbf{s}_n'[k]\mathbf{A}_{ni} = \sum_{j=1}^{n} \left(d_{ij}\mathbf{s}_j'[k]\mathbf{A}_{ji}\right). \qquad (2.2.2.1)$$

Therefore, if a site always selects itself as its determining state, it ends up operating as a standard Markov chain that is not influenced by its neighbors. It evolves according to $\mathbf{p}_i'[k+1] = \mathbf{s}_i'\mathbf{A}_{ii}$ at time step k where $\mathbf{A}_{ii}$ is a square stochastic matrix.

We can now list the parameters that define an IM:

1. The number of sites in the network, n.

2. The number of statuses that each site can take, m_i. We number the m_i possible statuses that site i can take, and we define c_{ij} to be the jth status that site i can take, where $1 \leq j \leq m_i$.

3. The probabilities d_{ij} with which determining neighbors are selected. These define the network of an IM. We assemble these probabilities in an $n \times n$ stochastic matrix $\mathbf{D} = \{d_{ij}\}$, which we call the *network influence matrix.*

4. The set of *state transition matrices* $\{\mathbf{A}_{ij}\}$, one for each pair of sites i and j.

5. We assume an initial distribution for the status at time 1 for each site. We define $\pi_i[j]$ as

$$\pi_i[j] = P(\mathbf{s}_i'[0] = c_{ij}), \tag{2.2.2.2}$$

   i.e., the probability that site i will take the status c_{ij} at time 0. We assemble these probabilities for each site i in a length-m_i vector $\pi_\mathbf{i}$, where the jth entry of $\pi_\mathbf{i}$ is equal to $\pi_i[j]$.

We use the notation $\lambda = (\mathbf{D}, \{\mathbf{A}_{ij}\}, \{\pi_\mathbf{i}\})$ to denote the parameter set of an IM: the influence matrix, the set of state transition matrices $\{\mathbf{A}_{ij}\}$ (one matrix for each pair of sites), and the set of vectors $\{\pi_\mathbf{i}\}$ (one vector per site) with the initial distribution for each site.

To construct a graphical representation of the network, we recall the common association of a directed graph $\Gamma[X]$ with an $n \times n$ matrix X: this graph has n nodes and a directed

edge of weight x_{ij} from node i to node j if and only if the entry x_{ij} of X is nonzero. For the purposes of the IM, it is more natural to use the graph of $\mathbf{D}'$, namely $\Gamma[\mathbf{D}']$, which is called the *network graph*. This graph has a directed edge weight of d_{ij} *from* site j *to* site i precisely when $d_{ij} > 0$, and the weight of this graph is the probability that site i will select site j to determine its next status. The sum of the weights of the inwardly directed edges into a node is therefore always 1.

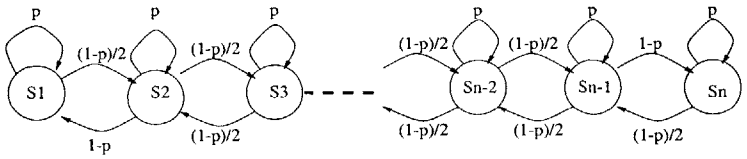

Figure 2-1: Graph corresponding to the matrix $\mathbf{L}$.

For example, a system with an influence matrix of the form

$$\mathbf{L} = \begin{bmatrix} p & 1-p & 0 & \dots & 0 & 0 & 0 \\ \frac{1-p}{2} & p & \frac{1-p}{2} & \dots & 0 & 0 & 0 \\ 0 & \frac{1-p}{2} & p & \frac{1-p}{2} & 0 & \dots & 0 \\ 0 & 0 & \frac{1-p}{2} & \ddots & \ddots & \dots & 0 \\ 0 & 0 & \dots & \ddots & p & \frac{1-p}{2} & 0 \\ 0 & 0 & 0 & \dots & \frac{1-p}{2} & p & \frac{1-p}{2} \\ 0 & 0 & 0 & 0 & \dots & 1-p & p \end{bmatrix}, \tag{2.2.2.3}$$

will have the network graph shown in the Figure 2-1 (i.e. a linear chain with n sites). Similarly, a system with an influence matrix of the form

$$\mathbf{C} = \begin{bmatrix} p & \frac{1-p}{2} & 0 & \dots & 0 & 0 & \frac{1-p}{2} \\ \frac{1-p}{2} & p & \frac{1-p}{2} & \dots & 0 & 0 & 0 \\ 0 & \frac{1-p}{2} & p & \frac{1-p}{2} & 0 & \dots & 0 \\ 0 & 0 & \frac{1-p}{2} & \ddots & \ddots & \dots & 0 \\ 0 & 0 & \dots & \ddots & p & \frac{1-p}{2} & 0 \\ 0 & 0 & 0 & \dots & \frac{1-p}{2} & p & \frac{1-p}{2} \\ \frac{1-p}{2} & 0 & 0 & 0 & \dots & \frac{1-p}{2} & p \end{bmatrix}, \tag{2.2.2.4}$$

corresponds to a system whose graph is a circular chain with n sites. An example of such a system, with eight sites, is shown in Figure 2-2.

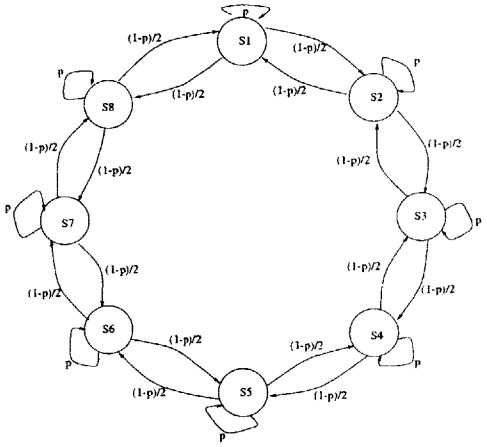

Figure 2-2: Graph corresponding to the matrix $\mathbf{C}$.

To describe the evolution of the IM more compactly, the status vectors and the next-status probability vectors for the n sites are stacked up into the "Multi Status" or state vector and the next-state "Multi Probability" or probability vector, respectively:

$$\mathbf{s}[k] = \begin{bmatrix} \mathbf{s}_1[k] \\ \vdots \\ \mathbf{s}_n[k] \end{bmatrix} \qquad \mathbf{p}[k+1] = \begin{bmatrix} \mathbf{p}_1[k+1] \\ \vdots \\ \mathbf{p}_n[k+1] \end{bmatrix}. \tag{2.2.2.5}$$

The IM is evolved by first calculating the next-state probability vector $\mathbf{p}[k+1]$ using the current state $\mathbf{s}[k]$, and then realizing the next state $\mathbf{s}[k+1]$ according to the probability vector $\mathbf{p}[k+1]$. If we assemble (2.2.2.1) for all sites, we can write

$$\mathbf{p}'[k+1] = \mathbf{s}'[k]\mathbf{H}, \tag{2.2.2.6}$$

where

$$\mathbf{H} = \begin{bmatrix} d_{11}\mathbf{A}_{11} & \dots & d_{n1}\mathbf{A}_{1n} \\ \vdots & & \vdots \\ d_{1n}\mathbf{A}_{n1} & \dots & d_{nn}\mathbf{A}_{nn} \end{bmatrix} = \mathbf{D}' \otimes \{\mathbf{A}_{ij}\}. \tag{2.2.2.7}$$

After $\mathbf{p}'[k+1]$ is determined as in (2.2.2.6), the state vector at time $k+1$ can be determined by independently realizing the next status of each site according to the probability vector in its portion of the Multi Probability vector $\mathbf{p}'[k+1]$. This process is denoted as

$$\mathbf{s}'[k+1] = MultiRealize(\mathbf{p}'[k+1]). \tag{2.2.2.8}$$

The notation used for $\mathbf{H}$ is the one defined on [2]. It is a generalization of the Kronecker product notation for a pair of matrices. Specifically, the $(i,j)th$ block in $\mathbf{H}$ is the result of using the ith row, jth column entry of the matrix $\mathbf{D}'$ (hence d_{ij}) to multiply the matrix $\mathbf{A}_{ij}$. The matrix $\mathbf{H}$ is not generally stochastic, but some of its properties are similar to those of stochastic matrices, as explained in [1].

2.2.1 Evolution of the Expected State of the Network

Given an initial state vector $\mathbf{s}'[0]$ for the network, we are interested in finding a recursion for the conditional expectation $E\big(\mathbf{s}'[k]|\mathbf{s}[0]\big)$. To do this, the conditional expectation $E\big(\mathbf{s}'[k+1]|\mathbf{s}[k]\big)$ is found first. Since the statuses at time $k+1$ are realized from the probabilities in $\mathbf{p}'[k+1]$, we have that

$$E\big(\mathbf{s}'[k+1]|\mathbf{s}[k]\big) = \mathbf{p}'[k+1] = \mathbf{s}'[k]\mathbf{H}, \tag{2.2.2.9}$$

where (2.2.2.6) was used in the second inequality. Applying this expectation repeatedly, we find that

$$E\big(\mathbf{s}'[k+1]|\mathbf{s}[0]\big) = \mathbf{s}'[0]\mathbf{H}^{k+1} = E\big(\mathbf{s}'[k]|\mathbf{s}[0]\big)\mathbf{H}. \tag{2.2.2.10}$$

2.2.2 The Homogeneous Influence Model

The IM can be specialized if we assume that each site takes the same number of possible status, $m_i = m$ for all i, and that $\mathbf{A}_{ij} = \mathbf{A}$ for all i, j. The evolution equation of the model is still given by (2.2.2.6), and (2.2.2.8), but with $\mathbf{H}$ now simply given by

$$\mathbf{H} = \begin{bmatrix} d_{11}\mathbf{A} & \dots & d_{n1}\mathbf{A} \\ \vdots & & \vdots \\ d_{1n}\mathbf{A} & \dots & d_{nn}\mathbf{A} \end{bmatrix} = \mathbf{D}' \otimes \mathbf{A}. \tag{2.2.2.11}$$

In this notation $\otimes$ denotes the standard Kronecker product. This specialization is called the *homogeneous influence model*, and will provide the starting ground for formulating the estimation problems this thesis aims to solve.

A Different Form

The homogeneous IM can be rewritten so that the evolution equations are in matrix-matrix form, instead of the vector-matrix form presented above. To do this, we first recall that the status of site i at time k is represented by a length-m status vector that contains a 1 in the position corresponding to the present status and 0 in all its other entries:

$$\mathbf{s}_i'[k] = [0 \dots 0\ 1\ 0 \dots 0].$$

This interpretation is the same as the one before. Likewise, we let $\mathbf{p}_i'[k]$ denote the PMF vector governing the status of site i at time k. We let the *network state matrix*, or simply the *state matrix*, $\mathbf{S}[k]$ and the *network probability matrix* $\mathbf{P}[k]$ to be defined as follows:

$$\mathbf{S}[k] = \begin{bmatrix} \mathbf{s}'_1[k] \\ \vdots \\ \mathbf{s}'_n[k] \end{bmatrix} \qquad \mathbf{P}[k+1] = \begin{bmatrix} \mathbf{p}'_1[k+1] \\ \vdots \\ \mathbf{p}'_n[k+1] \end{bmatrix}. \tag{2.2.2.12}$$

These definitions let us write the evolution equations for the homogeneous IM as:

$$\mathbf{P}[k+1] = \mathbf{D}\mathbf{S}[k]\mathbf{A} \tag{2.2.2.13}$$

$$\mathbf{S}[k+1] = MultiRealize\big(\mathbf{P}[k+1]\big) \tag{2.2.2.14}$$

$$E\big[\mathbf{S}[k]\big|\mathbf{S}[0]\big] = \mathbf{D}^k\mathbf{S}[0]\mathbf{A}^k \tag{2.2.2.15}$$

where the $MultiRealize()$ operation performs a random realization for each row of $\mathbf{P}[k+1]$.

Considering each row of $\mathbf{P}[k+1]$ separately helps to get an intuitive understanding of the process:

$$\mathbf{p}'_i[k+1] = d_{i1}\mathbf{s}'_1[k]\mathbf{A} + \ldots + d_{in}\mathbf{s}'_n[k]\mathbf{A} = \sum_{j=1}^{n} \big(d_{ij}\mathbf{s}'_j[k]\big)\mathbf{A}.$$

This whole section provides the background that is needed in the IM for the development of this thesis.

2.3 The Partially Observable Influence Model

Now that we have introduced the HMMs and the IM, we can introduce the POIM and formulate the problems that this thesis studies. A POIM is just an IM where only partial information of the state of the network is available at each time. Specifically, we shall assume that the status of a fixed set of nodes, the *observed nodes*, is known at each time. Just like in the HMM case, where the physical output of the system is a sequence of observations, in a POIM we have an IM that is evolving as described in Section 2.2, but where the state of the system is known only partially. At each time step t we see an observation, which consists of the state of a fixed set of sites. From these observations, we wish to extract information about the whole network.

At this point we introduce the basic notation we will use to describe a POIM. We denote by O the set of *observed sites*, and by U the set of *unobserved sites* in the network. The set of all sites of the network is denoted by S. We arbitrarily enumerate all the states in which these sets can be at each time, and write $O^t = i$ to denote that the set of observed sites at time t was in its ith state; or more simply, that the observation at time t is equal to i. For any other set of sites in the network, call it A, we denote by $A[k] = i$ that set A is in its state i at time k. Recall from Section 2.2 that m_j is the number of statuses that site j may take, so that the number of states that the set of observed sites may take is given by $\prod_{j \in O} m_j$ (i.e., in $O^t = i$ as above, $1 \leq i \leq \prod_{j \in O} m_j$). We denote a sequence of observations as $O^{l,k} = O^l O^{l+1} \dots O^k$, so the sequence of observations from time 0 to time T is written as $O^{0,T}$. We use similar notation for the set of unobserved sites U, or for any other set of sites we wish to describe.

To completely characterize a POIM, we then need to characterize the IM that underlies it, plus the set of observed sites O. We use the notation $\lambda = (\mathbf{D}, \{\mathbf{A}_{ij}\}, O)$ to denote the parameter set of a POIM.

A particular instance of a POIM is shown in Figure 2-3. The shaded sites are the observed ones ($O = \{2, 3\}$) and the rest are the unobserved sites ($U = \{1, 4, 5\}$). Therefore O will be

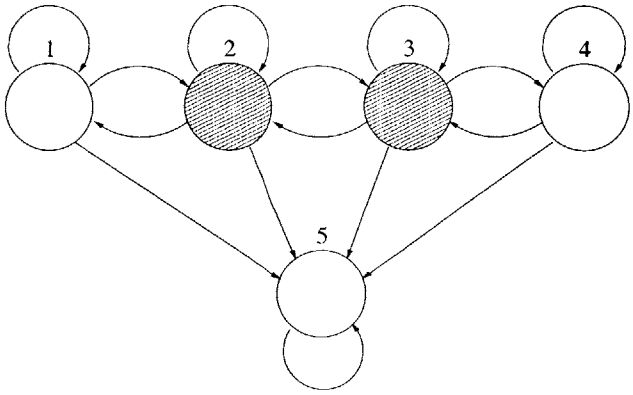

Figure 2-3: This IM with 5 sites is a POIM. The set of observed sites consists of sites 2 and 3, and the rest are the unobserved sites.

in one of $m_2 m_3$ possible states at each time, and U in one of $m_1 m_4 m_5$ states. The whole network S can take $m_1 m_2 m_3 m_4 m_5$ different states. This example will be carried along in the thesis to illustrate how the different techniques that we develop may be used.

The Homogeneous POIM

In some cases we will specialize this setting by assuming the IM that characterizes a POIM is homogeneous, with parameters $\lambda = (\mathbf{D}, \mathbf{A}, \{\pi_i\})$, and with n sites and m possible statuses per site (see Section 2.2). Recall that the state at time t of such an IM can be described by the state matrix $\mathbf{S}[t]$ and its corresponding dynamic equations, as defined in Section 2.2.2.

Relation between a HMM and a POIM

An HMM can be thought of as a POIM with two sites, where one of the sites, the unobserved site, is influenced only by itself and represents the underlying Markov chain; the second site is the observed one and always get influenced by the first. The status of the second site at each time can be thought of as the symbol emitted by the HMM. This is illustrated in Figure 2-4. In this sense, the POIM is a generalization of a HMM. It is also true that a POIM can be written as a large HMM, but the particular structure of the POIM is obscured by such an embedding.

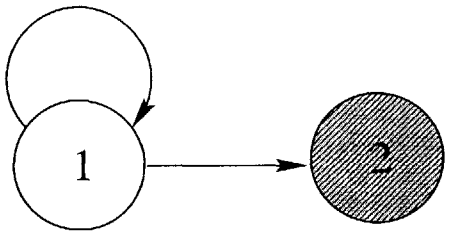

Figure 2-4: Graph of an IM that is equivalent to a HMM. Site 1 evolves like a Markov chain since it is always influenced only by itself. Site 2 accounts for the observation, which depends probabilistically only on the status of Site 1.

It can be seen from the discussion of the IM on Section 2.2 (see Equation 2.2.2.1) that the next-status probability vectors for Sites 1 and 2 are given by:

$$\begin{aligned} \mathbf{p}_1'[k+1] &= \mathbf{s}_1'[k]\mathbf{A}_{11} \\ \mathbf{p}_2'[k+1] &= \mathbf{s}_1'[k]\mathbf{A}_{12}. \end{aligned}$$

The first equation characterizes the Markov chain of site 1 while the second one describes the probabilistic dependence of the observation on the status of the first site.

Now we proceed to define the problems that this thesis addresses, which are in the context of the POIM.

2.4 The Three Problems of Study

The three main problems that are discussed in this thesis are the following:

1. Given a POIM λ and an observation sequence $O^{0,T}$, how do we efficiently calculate $P(O^{0,T}|\lambda)$, the likelihood of the observation sequence given the model?

2. Given an observation sequence $O^{0,T}$, and the POIM λ, how do we choose the state sequence $\mathbf{S}^{0,T}$ that *best* explains the observations?

3. How do we adjust the parameters of the underlying IM $\lambda = (\mathbf{D}, \mathbf{A}, \{\pi_i\})$ so as to maximize the probability of a sequence of observations, i.e., to maximize $P(O^{0,T}|\lambda)$?

Problem 1 evaluates how well the model λ matches a sequence of observations. For example, if there are different candidate IMs associated with a POIM we are observing, we may wish to solve for the likelihood of the observations given each of the models, and choose the model for which $P(O^{0,T}|\lambda)$ is the greatest. We refer to $L = P(O^{0,T}|\lambda)$ as the likelihood of the model.

Problem 2 tries to uncover the state of the network at the times when the observations were made. As in the HMM case, we may take different approaches to this problem. One would be to find the state $\mathbf{S}[k]$ of the network, at each time k, such that the probability that the network was at that state given the observation sequence is a maximum. That is:

$$\mathbf{S}[k] = \arg \max_{\text{all network states } i} P\big(\mathbf{S}[k] = i|O^{0,T}\big).$$

Finding such a solution might not be ideal, since the resulting sequence of states might not be a valid one, i.e., the resulting sequence of states might be one with invalid state transitions. Another approach would be to find the single sequence of states $\mathbf{S}^{0,T}$ that is most probable.

Problem 3 tries to adjust the parameters of the model so that the model can account for the observations better. The solution should be a method that trains the IM associated with a POIM, based on a sequence of observations, so that future observations can be accounted for in a better way. This problem can be thought of as a system identification problem.

The following chapters of this thesis develop methods to solve these three problems. We present exact answers for the first two problems. Unfortunately, we were only able to find an approximate solution to the third one that works only for the case of a homogeneous POIM.

Chapter 3

Solution to the Three POIM Related Problems

In this Chapter we present a general way to solve the three problems stated in Section 2.4. For convenience, we re-state them here:

1. Given a POIM λ and an observation sequence $O^{0,T}$, how do we efficiently calculate $P(O^{0,T}|\lambda)$, the likelihood of the observation sequence given the model?

2. Given an observation sequence $O^{0,T}$, and the POIM λ, how do we choose the state sequence $\mathbf{S}^{0,T}$ that *best* explains the observations?

3. How do we adjust the parameters of the underlying IM $\lambda = (\mathbf{D}, \mathbf{A}, \{\pi_i\})$ (assuming it is a homogeneous one) so as to maximize the probability of a sequence of observations, (i.e., to maximize $P(O^{0,T}|\lambda)$?

Unless explicitly mentioned, we assume we are dealing with an arbitrary POIM with parameters $\lambda = (\mathbf{D}, \{\mathbf{A}_{ij}\}, \{\pi_i\}, O)$, with n sites, and m_i possible statuses for site i. At each time k, we observe the status of the n_o observed sites, and we have n_u unobserved sites so that $n = n_o + n_u$. The methods presented in this Chapter would also hold if the set of observed sites changed with time. There are two reasons why we do not show that here. The first is that having a time-varying set of observed sites adds to the notation we use. Furthermore, in Chapter 4 we discuss how to take advantage of the graph structure of the

network to make the methods presented here more efficient. And this will only hold for the case when the set of observed sites is time-invariant.

In terms of notation, in the remainder of the thesis we will be writing a lot of expressions that involve probabilities of different sets of sites in the network being in different states. For example, we may wish to write the probability that set A is in state j at time k and in state l at time $k+1$. When space permits, we will write this as $P(A[k]=j, A[k+1]=l)$. However, if an expression such as the last one involves many sets of sites, we may write is as $P(A[k]=j, A[k+1]=l)$, implying that both $A[k]$ and $A[k+1]$ have a particular state.

The solutions we present are analogous to those used in the HMM scenario, described in Section 2.1. By this we mean that we use the same algorithms (Forward-backward, Viterbi, and Baum-Welch), but in a different context. Lastly before proceeding, we list some observations that will be used in the rest of the thesis.

1. An IM can be thought of as a giant Markov chain that can take $N' = \prod_{i \in S} m_i$ possible states, where S is the set of all sites in the network. One can think of enumerating these states and having the associated transition probabilities

$$\begin{aligned} p'_{ij} &= P(S[k+1]=j | S[k]=i) \\ &= \prod_{l=1}^{n} P(s_l[k+1]=j_l | S[k]) \\ &= \prod_{l=1}^{n} \Big(\sum_{r=1}^{n} d_{lr} \mathbf{s}'_r[k] \mathbf{A}_{rl} \mathbf{1}_{j_l} \Big), \ 1 \le i,j \le N'. \end{aligned} \tag{3.0.3.1}$$

   Here $s_l[k+1]=j_l$ means that site number j is in the status j_l which is consistent with the overall state j of the network, and the symbol $\mathbf{1}_{j_l}$ is a column vector of size m_l with a single entry of one in the j_lth position and otherwise filled with zeros. Equation 2.2.2.1 was used in the last equality.

2. Since we know the actual state of some part of the network at each time k, the number of states that the network can be in is not given by N' as above, but by $N = \prod_{i \in U} m_i$. This means that, given we know the state of the observed sites O, then the state space

of the network at k can be significantly smaller in size than N'. We may thus think of a POIM as a giant Markov chain with a state space consisting of the different states that the set of unobserved sites U may take and with time-varying transition probabilities. The transition probabilities are defined as follows:

$$\begin{aligned} p_{ij} &= P(U[k+1] = j | U[k] = i,\ O^k) \\ &= P(U[k+1] = j | S[k]) \\ &= \prod_{l \in U} P(s_l[k+1] = j_l | S[k]) \\ &= \prod_{l \in U} \Big(\sum_{r=1}^{n} d_{lr} \mathbf{s}'_r[k] \mathbf{A}_{rl} \mathbf{1}_{j_l} \Big), \qquad for\ 1 \leq i, j \leq N. \end{aligned} \tag{3.0.3.2}$$

Here j_l represents the status of site l which is consistent with the overall state j of the set of unobserved sites U. The dependence of p_{ij} on time comes through O^k, the observation at that time. This observation is the key to the methods we present.

3. In an influence model, the probability distribution used by each site to determine its next status, given the present state of the system, is independent of those of other sites. In fact, we note that to calculate the next-step probabilities for an arbitrary set of sites (say it's set A) we only need to know the state at the present time, of the sites that may influence any of the sites in A directly, and not the whole present state of the network. Therefore, if we group the set of sites that may influence the sites in A in another set B; we have that

$$\begin{aligned} P(A[k+1] = j | S[k] = i) &= P(A[k+1] = j | B[k] = i_b) \\ &= \prod_{l \in A} P(s_l[k+1] = j_l | B[k] = i_b) \\ &= \prod_{l \in A} \Big(\sum_{r \in B} d_{lr} \mathbf{s}'_r[k] \mathbf{A}_{rl} \mathbf{1}_{j_l} \Big), \end{aligned} \tag{3.0.3.3}$$

where j, i, and i_b can go from 1 to $\prod_{l \in A} m_i$, $\prod_{l \in S} m_i$, and $\prod_{l \in B} m_i$ respectively. Here j_l is the status of site l that is consistent with the set of sites A being in state i.

3.1 Problem 1: Finding the Probability of an Observation Sequence

We apply the Forward-backward algorithm to find the likelihood of the observation sequence. For this purpose, we enumerate the states that the set of unobserved sites U may take, and define the *forward* and *backward variables* (α and β, respectively), as follows:

$$\alpha_i[k] = P\big(U[k] = i, O^{0,k}\big), \qquad \text{and} \tag{3.1.3.1}$$

$$\beta_i[k] = P\big(O^{k+1,T} \big| U[k] = i,\ O^k\big). \tag{3.1.3.2}$$

These variables, once calculated for all times, allow us to compute the probability of an observation sequence $P(O^{0,T}|\lambda)$. This is because

$$\begin{aligned}
\alpha_i[k]\beta_i[k] &= P\big(U[k] = i,\ O^{0,k}\big)P\big(O^{k+1,T}\big|U[k] = i,\ O^k\big) \\
&= P\big(U[k] = i,\ O^k\big)P\big(O^{0,k-1}\big|U[k] = i,\ O^k\big)P\big(O^{k+1,T}\big|U[k] = i,\ O^k\big) \\
&= P\big(U[k] = i,\ O^k\big)P\big(O^{0,k-1}\big|U[k] = i,\ O^k\big)P\big(O^{k+1,T}\big|U[k] = i,\ O^k,\ O^{0,k-1}\big) \\
&= P\big(U[k] = i,\ O^k\big)P\big(O^{0,k-1},\ O^{k+1,T}\big|U[k] = i,\ O^k\big) \\
&= P\big(O^{0,T},\ U[k] = i\big),
\end{aligned}$$

so that the probability of an observation sequence may be calculated by

$$P(O^T|\lambda) = \sum_{i=1}^{N} \alpha_i[k]\beta_i[k].$$

There will be T different ways to calculate the likelihood of an observation provided we have the forward and backward variables for all times. N is the size of the state space of the unobserved sites U as discussed above.

3.1.1 The Forward-backward Algorithm for the POIM

To calculate the forward and backward variables, we use the Forward-backward Algorithm on the POIM, as follows:

1. Initialization:

$$\alpha_i[0] = P(U[0] = i, O^0) = \prod_{j=1}^{n} \pi_j[i_j] \qquad \text{for } 1 \leq i \leq N,$$

$$\beta_i[T] = 1 \qquad \text{for } 1 \leq i \leq N.$$

Here, i_j denotes the status of site j that is consistent with the state of the network defined by $U[0] = i$ and O^0.

2. Iteration:

$$\alpha_i[k+1] = \sum_{j=1}^{N} \alpha_j[k]\Phi(j,i) \qquad 1 \leq i \leq N \text{ and } 1 \leq t \leq T, \tag{3.1.3.3}$$

$$\beta_i[k] = \sum_{j=1}^{N} \Phi(i,j)\beta_j[k+1] \qquad 1 \leq i \leq N \text{ and } T-1 \geq t \geq 0, \tag{3.1.3.4}$$

where the function $\Phi(.,.)$ is defined as follows:

$$\Phi(i,j) = P(U[k+1] = j,\ O^{k+1} \mid U[k] = i,\ O^k) = \prod_{l=1}^{n} \Big(\sum_{r=1}^{n} d_{lr} \mathbf{s}'_r[k] \mathbf{A}_{rl} \mathbf{1}_{j_l} \Big),$$

for $1 \leq i, j \leq N$. In this equation j_l is the status of site l that is consistent with the state of the network defined by $U[k+1] = j$ and O^{k+1}.

3. Termination:

$$P(O^{0,T}|\lambda) = \sum_{i=1}^{N} \alpha_i[k]\beta_i[k] \quad \text{for any k.} \tag{3.1.3.5}$$

Proof

The initialization for the forward variables follows from the definition of a POIM. The initialization for the backward variables follows from the fact that the probability of an empty

event is zero:

$$\begin{aligned}\beta_i[T] &= P(O^{T+1,T}|U[k]=i,O^T)\\ &= P(\emptyset|U[k]=i,O^T)\\ &= 1.\end{aligned}$$

Now we show why the recursions in Equations 3.1.3.3 and 3.1.3.4 above hold:

For the forward variables we have that

$$\begin{aligned}\alpha_i[k+1] &= P\big(U[k+1]=i,O^{0,k+1}\big)\\ &= \sum_{j=1}^{N} P\big(U[k+1]=i,U[k]=j,O^{0,k},O^{k+1}\big)\\ &= \sum_{j=1}^{N} P\big(U[k+1]=i,O^{k+1}\big|U[k]=j,O^k\big)P\big(U[k]=j,O^{0,k}\big)\\ &= \sum_{j=1}^{N}\Phi(j,i)\alpha_j[k].\end{aligned}$$

Similarly, for the backward variables we have that

$$\begin{aligned}\beta_i[k] &= P\big(O^{k+1,T}\big|U[k]=i,\ O^k\big)\\ &= \sum_{j=1}^{N} P\big(O^{k+1},O^{k+2,T},U[k+1]=j\big|\ U[k]=i,\ O^k\big)\\ &= \sum_{j=1}^{N} P\big(O^{k+1},U[k+1]=j\big|\ U[k]=i,\ O^k\big)P\big(O^{k+2,T}\big|O^{k+1},U[k+1]=j,U[k]=i,O^k\big)\\ &= \sum_{j=1}^{N} P\big(U[k+1]=j,\ O^{k+1}\big|\ U[k]=i,\ O^k\big)P\big(O^{k+2,T}\big|U[k+1]=j,\ O^{k+1}\big)\\ &= \sum_{j=1}^{N}\Phi(i,j)\beta_j[k+1].\end{aligned}$$

Computation

The number of computations it takes to do the initialization of the algorithm is on the order of $Nn+N$, since each forward variable is a product of n terms, and each backward variable is equal to 1. Assuming that evaluating the product $\mathbf{s}'_r[k]\mathbf{A}_{rl}\mathbf{1}_{j_l}$ takes constant time to evaluate (quite possible since what this product does is extract a single entry from a matrix), and that $d_{ij} \neq 0$ for all i, j, then evaluating $\Phi(i, j)$ takes on the order of n^2 computations. Recall n is the number of sites in the network. Then it takes on the order of Nn^2 computations to calculate each forward or backward variable. Since there are $2N$ forward and backward variables per time step, finding them takes on the order of Tn^2N^2 computations. Finally the termination step will require a number of computations on the order of N. Therefore the running time R of the algorithm is dominated by the recursions in step 2 and is given by $R = N^2n^2T$. Plugging in for $N = \prod_{i\in U} m_i$, we see that

$$R = Tn^2(\prod_{i\in U} m_i^2).$$

Therefore, we see that the running time increases horribly, not with the number of sites in the network, but with the number of unobserved sites, and the number of statuses they may take. For example,for a homogeneous POIM, since all sites may take m statuses, the running time becomes $R = Tn^2m^{2n_u}$, which is exponential in the number of unobserved sites. This makes the use of this algorithm useful only for POIMs with a small number of unobserved sites and/or with sites that may take only a few number of statuses. One good thing is that it is linear in time so that doubling the length of the time interval just doubles the time the algorithm takes.

The story for the memory required to run this algorithm is similar. There are a total of $2TN$ forward and backward variables that need to be stored, as well as the model parameters λ.

The running time and memory of this algorithm usually explode as the number of sites gets large. However, as will be discussed later, this is the general version of the algorithm

that will work for any POIM. There are POIMS with network structure that allow the use of more efficient ways to calculate the forward and backward variables. We will focus on these issues of network structure in Chapter 4.

3.1.2 Example

To illustrate how this algorithm may be used, consider the homogeneous POIM shown in Figure 3-1. We will refer to this POIM repeatedly to illustrate the different methods discussed in this thesis.

Each site can take one of two statuses ($m = 2$), denoted by 1 and 2, and the model parameters are:

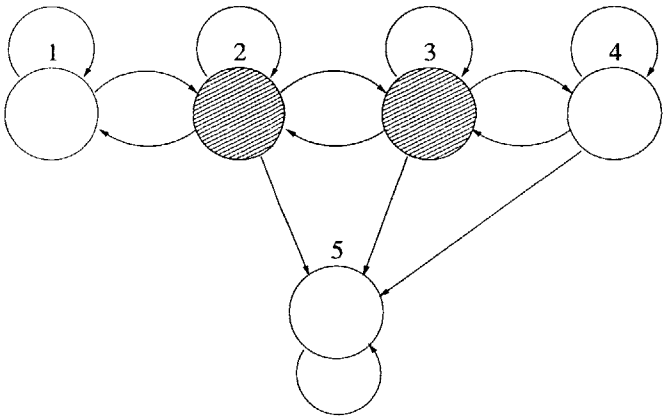

Figure 3-1: Homogeneous POIM with 5 sites. We will use this network as an example of how to use the algorithms developed in this thesis.

$$
\begin{aligned}
\mathbf{D} &= \begin{bmatrix} .5 & .5 & 0 & 0 & 0 \\ .25 & .5 & .25 & 0 & 0 \\ 0 & .25 & .5 & .25 & 0 \\ 0 & 0 & .5 & .5 & 0 \\ 0 & .25 & .25 & .25 & .25 \end{bmatrix}, \\
\mathbf{A} &= \begin{bmatrix} p & 1-p \\ 1-q & q \end{bmatrix}, \text{ and} \\
\pi_i &= \begin{bmatrix} 0 & 1 \end{bmatrix} \text{ for all sites } 1 \leq i \leq 5.
\end{aligned}
$$

This is a POIM whose initial state is completely determined, with all sites being in status 2. Let O and U denote the set of observed and unobserved sites $O = \{2, 3\}$ and $U = \{1, 4, 5\}$. We wish to illustrate how to use the forward-backward algorithm to find the probability of an observation sequence. To make this example concrete, we let $p = q = .1$ so that $\mathbf{A}$ is determined. Assume we are watching such a POIM evolve in time for $T + 1$ time steps, and we see a given observation sequence $O^{0,T}$. To find $P(O^{0,T})$ we compute the forward and backward variables and then use Equation 3.1.3.5 for any time k between 0 and T.

We do this using MATLAB and find that, for example, the probability of an observation sequence where site 2 takes the following statuses $s_2 = 2121221222$ and where site 3 takes the statuses $s_3 = 2111211212$ is 0.7787×10^{-5}. Similarly, if we observe the system for three time steps only, and find that $s_2 = 212$ and $s_3 = 211$, then the probability of the observation is 0.0885. A sanity check for the forward and backward variables can be done by making sure Equation 3.1.3.5 yields the same $P(O^{0,T})$ for all times k between 0 and T.

3.2 Problem 2: Estimating the State of the Network Given the Observations

3.2.1 Local Decoding

The Forward-backward algorithm allows us to calculate the most likely state of the system at each time, given a sequence of observations, or

$$\arg \max_{1 \leq j \leq N} P\big(U[k] = j | O^{0,T}\big).$$

This is sometimes referred to as localized decoding.

To do this we first note that

$$P\big(U[k] = j | O^{0,T}\big) = \frac{P\big(U[k] = j, O^{0,T}\big)}{P\big(O^{0,T}\big)}$$

so that the same state $U^*[k]$ that maximizes $P\big(U[k] = j | O^{0,T}\big)$ maximizes $P\big(U[k] = j, O^{0,T}\big)$ too. Next, we note that this probability is calculated via the forward and backward variables by

$$\alpha_j[k]\beta_j[k] = P\big(O^{0,T}, U[k] = j\big),$$

as shown in Section 3.1.1. Therefore, we find the most likely state of the system at time k, denoted by $U^*[k]$, given the observations; by

$$U^*[k] \;=\; \arg \max_{1 \leq j \leq N} \alpha_j[k]\beta_j[k]. \tag{3.2.3.1}$$

If one is interested in doing localized decoding of an arbitrary set of unobserved nodes A, then once we find the probability distribution $P\big(O^{0,T}, U[k] = j\big)$, we may simply add over the unobserved sites that are not in A to find the probability distribution $P\big(O^{0,T}, A[k] = j\big)$. Then, we simply choose as out estimate for $A[k]$ the state j that maximizes $P\big(O^{0,T}, A[k] = j\big)$ at each time k.

In general, doing localized decoding may not be what is desired. One may wish to solve for the most likely sequence of states $U^{0,T}$, from time 0 to time T, given the observations. This is referred to as global decoding.

Computation

To do localized decoding one must perform the Forward-backward algorithm, which as discussed above, takes on the order of Tn^2N^2 computations. Then, for each time k, we need to evaluate Equation 3.2.3.1 which takes on the order of N computations. Therefore estimating $U^*[k]$ for all times between 0 and T takes on the order of Tn^2N^2 computations.

3.2.2 The Viterbi Algorithm: Global Decoding

Now we propose a way to find the most likely state sequence given the observations, or

$$U^{*0,T} = \arg \max_{\text{all paths } U^{0,T}} P(U^{0,T}|O^{0,T}).$$

The solution we propose is analogous to that used to find the most likely state sequence, given a sequence of observations, in a Hidden Markov Model and uses the Viterbi algorithm.

For this purpose we define the *Viterbi variables,* which are the highest probability for a sequence of states from time 0 to time k that accounts for the observations up to time k, and whose state at time k is $U[k] = i$:

$$\delta_i[k] = \max_{U^{0,k-1}} P\big(O^{0,k}, U^{0,k-1}, U[k] = i\big).$$

We will prove below that the Viterbi variables may be found recursively by

$$\delta_i[k+1] = \max_{1 \le j \le N} \delta_j[k]\Phi(j,i),$$

where $\Phi = P(U[k+1] = i, O^{k+1}|U[k] = j, O^k)$ just like in the previous section. Assuming this holds, we can then find the state sequence that maximizes the probability of the observation sequence, by picking the maximum $\delta_i[T]$ and backtracking the sequence of states that led to

it. For this purpose we keep track of the previous state in the path for each $\delta_i[k]$ storing it in the array $\psi_i[k]$. Then, we calculate $U^{*0,T}$ by the following algorithm, which we call the Viterbi algorithm on the POIM:

1. Initialization:

$$\begin{aligned} \delta_i[0] &= \prod_{j=1}^{n} \pi_j[i_j] \qquad \text{for } 1 \leq i \leq N \\ \psi_i[0] &= 0 \qquad \text{for } 1 \leq i \leq N. \end{aligned}$$

Here, i_j denotes the status of site j that is consistent with the state of the network defined by $U[0] = i$ and O^0.

2. Iteration:

$$\begin{aligned} \delta_i[k+1] &= \max_{1 \leq j \leq N} \delta_j[k]\Phi(j,i) \qquad \text{for } 1 \leq i \leq N \text{and } 1 \leq k \leq T \\ \psi_i[k+1] &= \arg\max_{1 \leq j \leq N} \delta_j[k]\Phi(j,i) \qquad \text{for } 1 \leq i \leq N \text{and } 1 \leq k \leq T, \end{aligned}$$

where $\Phi(j,i) = P(U[k+1] = i,\ O^{k+1} \big|\ U[k] = j,\ O^k)$.

3. Termination:

$$\begin{aligned} U^{*T} &= \arg\max_{1 \leq i \leq N} \delta_i[T] \\ U^{*k} &= \psi(U^{*k+1}) \qquad \text{for } T-1 \geq t \geq 0. \end{aligned}$$

Proof

The initialization follows directly from the definition of a POIM. Now we prove the recursion for the Viterbi variables:

$$\begin{aligned}
\delta_i[k+1] &= \max_{U^{0,k+1}} P\big(O^{0,k+1}, U^{0,k}, U[k+1]=i\big) \\
&= \max_{U^{0,k}} P\big(O^{0,k}, U^{0,k-1}, O^{k+1}, U[k]=j, U[k+1]=i\big) \\
&= \max_{U^{0,k}} P\big(O^{0,k}, U^{0,k-1}, U[k]=j\big) P\big(O^{k+1}, U[k+1]=i \big| U[k]=j, O^{0,k}, U^{0,k-1}\big) \\
&= \max_{U^{0,k}} P\big(O^{0,k}, U^{0,k-1}, U[k]=j\big) P\big(O^{k+1}, U[k+1]=i \big| U[k]=j, O^{k}\big) \\
&= \max_{U[k]=j} \Big(\max_{U^{0,k-1}} \big[P\big(O^{0,k}, U^{0,k-1}, U[k]=j\big)\big] P\big(O^{k+1}, U[k+1]=i \big| U[k]=j, O^{k}\big)\Big) \\
&= \max_{U[k]=j} \big(\delta_j[k]\Phi(j,i)\big) \\
&= \max_{1\le j\le N} \delta_j[k]\Phi(j,i).
\end{aligned}$$

Computation

The initialization step takes about Nn since each variable is a product of n terms and there are N variables to find. We assume, like we did for the Forward-backward algorithm, that evaluating the Φ function takes about n^2 computations. Then for the recursions in step 2, finding each Viterbi variable takes about Nn^2 computations, and there are N of them per time step. So for a sequence of observations of length T this will take on the order of Tn^2N^2 computations. The termination step then takes on the order of TN computations. The running time of this algorithm is thus dominated by the recursion in step 2 and is given by $R = Tn^2N^2$, which is the same as that for the Forward-backward algorithm.

3.2.3 Example

In this example, we do localized and global decoding with the homogeneous POIM of Section 3.1.2, shown in Figure 3-1. To do localized decoding, we compute the forward and backward variables and then use Equation 3.2.3.1 to generate our estimate for the state of the network at time k. Because there are 3 sites whose status is unknown in the network, and each may take 2 statuses, there are 8 possible network states at each time. We numbered

each of these states from 1 through 8 and did the estimates. The upper plot of the figure below shows one such estimate, done for a simulation with 50 time steps with parameters $p = q = .1$. The solid line represents the actual state of the network while the dotted line is the estimate obtained by localized decoding.

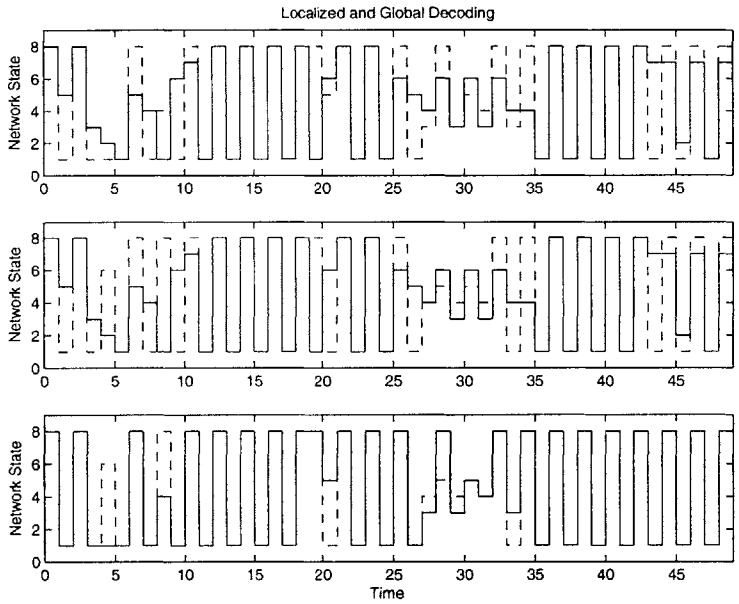

Figure 3-2: This figure illustrates localized and global decoding. In the upper two plots, the solid line represents the actual state of the network, while the dotted line is the estimate. The upper plot shows the result of doing localized decoding, the middle one shows the result of doing global decoding on the same data, while the lower plot shows the difference between the estimates obtained via these two methods.

For another particular run of the experiment, the estimator successfully predicted the state of the network in 201 time steps out of 500. This is not bad considering that an estimator that chooses one of the eight states at random at each time is expected to successfully predict the state of the network about 62 times out of the 500 time steps. For 480 independent runs of an experiment over 100 time steps, we did localized decoding of the state of the network and found that the estimator is successful 46.228 percent of the time (with a standard deviation of 6%).

If instead of estimating the whole state of the network, we were interested in estimating a single site, say site 5 then we would expect the number of hits (or successes) of our estimator to increase. This is exactly what happens. To do localized decoding for site 5, for example, we use the forward and backward variables to find the probability distribution $P\big(O^{0,T}, s_5[k] = j\big)$ and we choose as our estimate of $s_5[k]$ the state j that maximizes $P\big(O^{0,T}, s_5[k] = j\big)$. For 480 independent runs of an experiment over 100 time steps, we did localized decoding of the status of sites 1, 4 and 5 and found that the estimator is successful 78% of the times for site 1, 78% of the time for site 4 and 72% of the time for site 5. The standard deviation for the three sites is about 5%.

To do global decoding, we use the Viterbi algorithm described in this section. We find, for example, that if the observation sequence is given by $s_2 = 2212$ and $s_3 = 2122$, then the most likely state sequence for the network has the following site sequences: $s_1 = 2112$, $s_4 = 2121$ and $s_5 = 2121$ with a probability of 0.00086.

The middle plot in Figure 3-2 shows the estimate obtained via global decoding for the same data as the one used for the upper plot, where localized decoding is shown. The solid line represents the actual state of the network while the dotted line is the global decoding estimate. The estimates obtained via local and global decoding are different for the same data. This is shown in the lower plot of Figure 3-2. The global decoding estimate is shown as a dashed line while the local decoding estimate is the solid line.

3.3 Problem 3: Estimating the Model Parameters

In this section, we address the problem of finding the POIM λ^* that can best account for a sequence of observations, in the sense that

$$\lambda^* = \max_{\lambda} P(O^{0,T}|\lambda). \tag{3.3.3.1}$$

Assuming we now the number of sites in the network n and the set of observed sites O, we would like to find the rest of the parameters for the model that maximize the quantity above.

These parameters we would like to find are thus the influence matrix $\mathbf{D}$, the matrices $\{\mathbf{A}_{ij}\}$, and the initial distributions $\{\pi_{\mathbf{i}}\}$, as defined in Section 2.2.

This problem is hard, and we do not have a solution even for the case where all nodes are observed. This is described next.

3.3.1 Estimating the Paramters of an IM

If we assume that all nodes are observed, then given a sequence of observations we can write out the likelihood $P(O^{0,T}|\lambda)$ explicitly in terms of the parameters of the model. The problem of estimating the paramters then becomes an optimization problem where we want to find the value of $\mathbf{D}$, $\{\mathbf{A}_{ij}\}$, and $\{\pi_{\mathbf{i}}\}$ that maximize the likelihood, with the constraints that the matrices $\mathbf{D}$ and $\{\mathbf{A}_{ij}\}$ must be row stochastic, and that the vectors $\{\pi_{\mathbf{i}}\}$ are valid probablity distributions. Further studies need to be done to evaluate what would be appropriate optimization methods to address this problem.

3.3.2 Estimating the Paramters of an POIM

In Section 2.1.5 we saw that for the HMM case there is no exact solution for the analogous problem. There is only a method that comes up with a solution that maximizes the likelihood in Equation 3.3.3.1 locally, instead of globally. This method is called the Baum-Welch method and was discussed in Section 2.1.6. Unfortunately, the situation for the POIM is worse. The method we present is only useful for the homogeneous POIM. That is, it may only be used provided we assume that the $\{\mathbf{A}_{ij}\}$ matrices all all equal to a single matrix $\mathbf{A}$, as described in Section 2.2.2. Furthermore, it will only return the exact solution provided we start with a good enough estimate of the parameters.

Having said that, then our problem becomes that of finding the influence matrix $\mathbf{D}$, the matrix $\mathbf{A}$, and the initial distributions $\{\pi_{\mathbf{i}}\}$, that maximize Equation 3.3.3.1, with $\lambda = (\mathbf{D}, \mathbf{A}, \{\pi_{\mathbf{i}}\}, O)$.

The method we propose consists of the following steps, which will be explained in detail:

1. With an initial estimate of the POIM parameters **D**, **A**, and $\{\pi_{\mathbf{i}}\}$, calculate the transition probabilities p'_{ij} of the big Markov chain associated with it. This is done via Equation 3.0.3.1. Recall that this Markov chain has size $N' = \prod_{i=1}^{n} m_i$. Similarly find the initial distributions of this big Markov chain by

$$P(S[0] = i) = \prod_{j=1}^{n} \pi_{\mathbf{i}}[j_i],$$

where j_i denotes the status of site j that is consistent with the state i of the network.

2. Starting with the probabilistic description of the Markov chain from Step 1, apply the Baum-Welch algorithm (see Section 2.1.6) to find the transition probabilities p^*_{ij} and the initial distribution $P(S[0] = i)^*$ that maximize the likelihood of the observation sequence.

3. Recover the parameters of the POIM from p^*_{ij} and $P(S[0] = i)^*$, and check for consistency. What we mean by consistency will be clarified later. If the parameters found are consistent, or good enough, we terminate the procedure. Otherwise we repeat the process from Step 1 with a different initial estimate of the parameters **D**, **A**, and $\{\pi_{\mathbf{i}}\}$.

3.3.3 Step 1

For Step 1 we assume we have an initial estimate of the homogeneous POIM. For example we might know a good deal about the graph structure and know which entries in the influence matrix **D** are zero. This is reasonable provided we have an idea of the network topology. Then, we need to enumerate all the states that the network can take, and find the parameters of the Markov chain associated with it. These parameters consist of the N'^2 transition probabilities p'_{ij} found by Equation 3.0.3.1; plus the N' probabilities for the initial state of the system

$$P(S[0] = j) = \prod_{i=1}^{n} \pi_{\mathbf{i}}[j_i].$$

Calculating each p'_{ij} and $P(S[0] = j)$ in this way will take about n^2, and n computations respectively. Therefore the total number of computations for this step is about $N'^2 n^2 + N'n$.

3.3.4 Step 2: The Baum-Welch Algorithm

Step 2 consists of applying the Baum-Welch algorithm for the Markov chain found in Step 1. This method, however, will be slightly simplified from the one described in Section 2.1.6 because in this case, there are no emitted symbols. We therefore describe the Baum-Welch algorithm for this Markov chain.

Starting with parameters found in Step 1, at each iteration of the algorithm we find an improved estimate of these parameters. For this purpose at each iteration we must solve the Forward-backward algorithm as described in Section 3.1.1. Then we define the probability that the network is in state i at time k given the observations as

$$\gamma_i[k] = P(S[k] = i|O^{0,T}) = \frac{\alpha_i[k]\beta_i[k]}{P(O^{1,T})} = \frac{\alpha_i[k]\beta_i[k]}{\sum_{i=1}^{N} \alpha_i[k]\beta_i[k]}.$$

Note, however, that there are only N forward and backward variables per time step, but N' states that S may be in. What happens is that $N' - N$ states are not consistent with the observations at that time, and therefore the γ variable for those states is just zero.

We also define $\xi_{i,j}[k]$ as the probability that the network is in state i at time k and state j at time $k+1$, given the model and the observation sequence, i.e.,

$$\xi_{i,j}[k] = P(S[k] = i, S[k+1] = j|O^{1,T}).$$

Then we can write $\xi_{i,j}[k]$ as:

$$\begin{aligned} \xi_{i,j}[k] &= \frac{P(S[k] = i, S[k+1] = j, O^{1,T})}{P(O^{1,T})} \\ &= \frac{\alpha_i[k]p'_{ij}\beta_j[k+1]}{P(O^{1,T})} \\ &= \frac{\alpha_i[k]p'_{ij}\beta_j[k+1]}{\sum_{i=1}^{N} \alpha_i[k]\beta_i[k]}. \end{aligned} \tag{3.3.3.2}$$

And we can relate $\xi_{i,j}[k]$ and $\gamma_i[k]$ by

$$\gamma_i[k] = \sum_{j=1}^{N} \xi_{i,j}[k].$$

Using these equations, we find the improved model parameters $\overline{p'}_{ij}$ and $\overline{P(S[0]=i)}$ by:

$$\begin{aligned}
\overline{P(S[0]=i)} &= \text{expected frequency in state } i \text{ at } k=0 \\
&= \gamma_i[0] \qquad (3.3.3.3) \\
\overline{p'}_{ij} &= \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i} \\
&= \frac{\sum_{k=0}^{T-1} \xi_{i,j}[k]}{\sum_{k=0}^{T-1} \gamma_i[k]} \qquad (3.3.3.4)
\end{aligned}$$

This process is repeated until the new estimate and the old estimate are the same, at which point we will have found a local maxima in the likelihood function. We denote the final estimates by p^*_{ij} and $P(S[0]=i)^*$. The number of iterations for this algorithm to terminate is unknown. We just know that each estimate will be just as good or better than the previous one. Each iteration still takes a large number of computations. It takes about Tn^2N^2 computations to execute the Forward-backward algorithm, and about N more to find $P(O^{0,T})$. Having done this it will take about TN' computations to find the γ variables, and TN'^2 to find the ξ variables. Finally computing the new estimates will take about TN'^2. Therefore each iteration will take on the order of $T(n^2N^2 + N'^2)$ computations. This is horrendous, but we do not know how to do any better.

This formulation of the Baum-Welch is valid since it is just restating the algorithm in Section 2.1.6 for the case when there is only one emitted symbol independent of the state.

3.3.5 Step 3

In Step 3 we recover the influence matrix $\mathbf{D}$, the matrix $\mathbf{A}$, and the initial distributions $\{\pi_i\}$ from the Markov chain parameters p^*_{ij} and $P(S[0]=i)^*$.

First, assume that the parameters found in Step 2 are those of the associated Markov chain of an homogeneous POIM. Under this assumption, we show how to recover the POIM parameters.

First we find the initial distribution for each site by:

$$\pi_i[j] = P(s_i[0] = j) = \sum_{l|s_i[0]=j} P(S[0] = l)^*.$$

Next, we may find the i, j entry of the matrix $\mathbf{A}$, denoted by a_{ij}, by the following equation:

$$a_{ij} = \sqrt[n]{p'^*_{s_i s_j}},$$

where s_i (s_j) denotes the state of the network where all the sites are in status i (j). This follows from the fact that for a homogeneous POIM:

$$\begin{aligned} p'^*_{s_i s_j} &= P(S[k+1] = s_i | S[k] = s_j) \\ &= \prod_{l=1}^{n} \left(\sum_{r=1}^{n} d_{lr} \mathbf{s}'_r[k] \mathbf{A} \mathbf{1}_j \right) \\ &= \prod_{l=1}^{n} \left(\sum_{r=1}^{n} d_{lr} \mathbf{1}'_i \mathbf{A} \mathbf{1}_j \right) \\ &= \prod_{l=1}^{n} \left(\sum_{r=1}^{n} d_{lr} a_{ij} \right) \\ &= a_{ij}^n, \end{aligned}$$

where $\mathbf{1}_j$ denotes a length-m column vector with a single entry of 1 in its jth position, and filled with zeros.

Next, we show how to recover the influence matrix $\mathbf{D}$. We will explain how to find the i, jth entry of $\mathbf{D}$, or d_{ij} which is the probability that site i selects site j as its influencing site. For this purpose, we work with three different network states which we denote by S_1, S_2 and S_3. The state S_1 denotes the sate of the network where site j is in status b and where all other sites are in states a, with $a \neq b$. State S_2 is one where all the sites are in the same

status, say status c. State S_3 is equal to S_2 for all sites except for site i which is in status d, with $d \neq c$. The choice for the statuses a, b, c, and d is completely arbitrary as long as $a \neq b$ and $d \neq c$. To illustrate what we mean, let's assume we want to find d_{13} in a homogeneous POIM with $n = 4$, and where each site can take 4 different statuses. Say we choose our a, b, c, and d as the statuses 1, 2, 3, and 4, respectively, that each site may take. Then the influence matrix $\mathbf{S}$ for these network states would look like so:

$$\mathbf{S} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \text{ for } S_1,$$

$$\mathbf{S} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \text{ for } S_2, \text{ and}$$

$$\mathbf{S} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \text{ for } S_3.$$

Now, consider the probability that the network transitions from S_1 to S_2, and from S_1 to S_3.

$$\begin{aligned}
p_1 &= P(S[k+1] = S_2 | S[k] = S_1) \\
&= \prod_{l=1}^{n} P(s_l[k+1] = j_{S_2} | S[k] = S_1) \\
&= \prod_{l=1}^{n} \Big(\sum_{r=1}^{n} d_{lr} \mathbf{s}'_r[k] \mathbf{A} \mathbf{1}_{j_{S_2}} \Big) \\
&= \Big[\prod_{l=1, l \neq i}^{n} \Big(\sum_{r=1}^{n} d_{lr} \mathbf{s}'_r[k] \mathbf{A} \mathbf{1}_c \Big) \Big] ((1 - d_{ij}) a_{ac} + d_{ij} a_{bc}), \text{ and} \\
p_2 &= P(S[k+1] = S_3 | S[k] = S_1) \\
&= \prod_{l=1}^{n} P(s_l[k+1] = j_{S_3} | S[k] = S_1) \\
&= \prod_{l=1}^{n} \Big(\sum_{r=1}^{n} d_{lr} \mathbf{s}'_r[k] \mathbf{A} \mathbf{1}_{j_{S_3}} \Big) \\
&= \Big[\prod_{l=1, l \neq i}^{n} \Big(\sum_{r=1}^{n} d_{lr} \mathbf{s}'_r[k] \mathbf{A} \mathbf{1}_c \Big) \Big] ((1 - d_{ij}) a_{ad} + d_{ij} a_{bd}).
\end{aligned}$$

Here a_{ij} is the entry of the matrix $\mathbf{A}$ in row i and column j, and we denote by $s_l[k+1] = j_{S_2}$ that site l is in the status j_{S_2} which is consistent with the state of the network S_2 at time k. Therefore, we see that

$$\frac{p_1}{p_2} = \frac{(1 - d_{ij}) a_{a,c} + d_{ij} a_{bc}}{(1 - d_{ij}) a_{a,d} + d_{ij} a_{bd}},$$

and solving for d_{ij} we find that

$$d_{ij} = \frac{p_2 a_{ac} - p_1 a_{ad}}{(p_2 a_{ac} - p_1 a_{ad}) - (p_2 a_{bc} - p_1 a_{bd}).} \tag{3.3.3.5}$$

Therefore, to find each entry in the influence matrix, we choose some states S_1, S_2, and S_3 as described above, get their corresponding transition probabilities p_1 and p_2 from Step 2 and use Equation 3.3.3.5.

This method, however, returns a consistent answer only when the transition probabilities from Step 2 indeed correspond to a homogeneous POIM. Otherwise the value we will get for

d_{ij} will be different depending on our choice of the statuses a, b, c, and d. Therefore, to see if the POIM found by this method is consistent, we need to first make sure that all the d_{ij} are valid probability measures (positive and less than or equal to 1). If they are then we need to recompute the transition probabilities p_{ij} as in Step 1 for the big associated Markov chain. If they are equal to the probabilities found in Step 2 (if $p_{ij} = p^*_{ij}$ for all i, j), then we have found a valid solution. Otherwise we have a choice to make: we may start the process form Step 1 with a different initial estimate of the POIM, or we might find that the likelihood (i.e., $P(O^{0,T}|\lambda)$) of the one we found is already good enough and stay with it.

Assuming this step is successful, meaning that we check for the validity of the POIM and find it is valid, or good enough, then at least the following computations are required for Step 3:

- Finding each entry of the initial distributions π_i takes $N/m = m^{n-1}$ computations. There are mn entries, so this takes Nn computations.
- Finding the $\mathbf{A}$ matrix takes on the order of m^2 computations. Finding the $\mathbf{D}$ matrix takes at least n^2 computations.
- Verifying the validity of the POIM might take $N'^2 n^2$, to find the transition probabilities p'_{ij}, as described in Step 1.

Therefore the total number of computations is dominated by the validation step, which takes on the order of $N'n^2$ computations. The order of the total number of computations, for the three steps, is dominated by $N'^2(n^2 + T)$.

Chapter 4

Estimation by Exploiting Graph Structure on a POIM

The main goal of this chapter is to present methods that address the problems of finding the probability of an observation sequence and of state estimation in a POIM. These methods take advantage of the structure of the network graph and can be significantly more efficient than those presented in Chapter 3.

Recall from Section 2.2 that the network graph has a directed edge weight of d_{ij} from site j to site i precisely when $d_{ij} > 0$, and the weight of this graph is the probability that site i will select site j to determine its next status.

This chapter is organized as follows: First, we start by laying the foundations which let us take advantage of the graph structure. We introduce the concepts of ζ-independent sets, and of ζ-minimal sets and partitions. Then we discuss how to use these concepts to find the probability of an observation sequence in a POIM, followed by an efficient algorithm to estimate the present state of sets of sites in the network given the observations up to the present. Finally, we show how to use graph structure to find the forward and backward variables for the whole network.

Recall that the set of all sites in the network consists of the union of the observed and unobserved sites (i.e., $S = O \cup U$). In this chapter, it will be useful to further decompose the network into sets of sites with common characteristics. In particular, we classify the unobserved sites as *influencing* and *non-influencing*. An influencing site is an unobserved site from which there exists a path to at least one observed site. They are sites with the potential to eventually influence at least one observed site. Similarly, a non-influencing site is one from which there is no path to any observed site. That is, they are incapable of influencing any observed site. The effect of a non-influencing site is only local, in that it may only affect the future dynamics of other non-influencing sites. It will never have any effect either on an influencing or an observed site. We denote the set of influencing sites by I and that of non-influencing sites by $\bar{I}$, so that $U = I \cup \bar{I}$. The value of this classification will become evident later in this chapter.

To illustrate the previous definitions consider the network graph shown in Figure 4-1: Sites 1 and 4 are influencing, and site 5 is non-influencing.

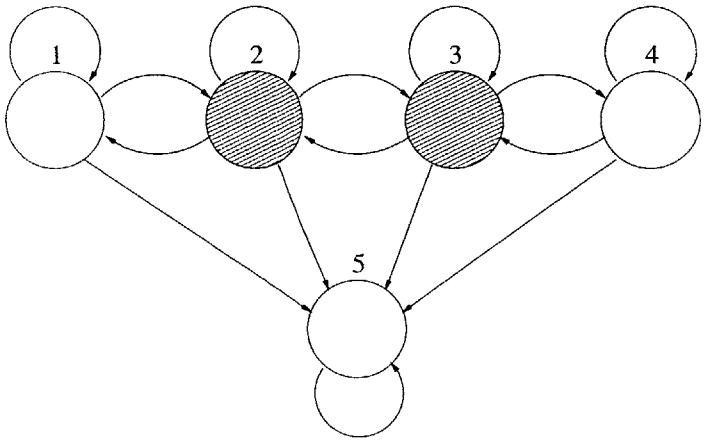

Figure 4-1: In this POIM the influencing sites are sites 1 and 4 while site 5 is the non-influencing.

4.1 The ζ-Variables and Related Concepts

There are sets of sites in the network whose probability distributions at time k given the observations up to that time, may be calculated independently of the rest of the network. We call these probabilities ζ-variables and these sets ζ-independent sets. We study these sets and these variables, and develop some further concepts related to them that will be useful later in the Chapter.

In this chapter, we will make use of some basic notions from set theory. We refer to a collection of nodes from the network graph $\Gamma[\mathbf{D}']$ as a set. A subset B of a set A is a set with elements that are all in A (i.e.$\forall x \in B, x \in A$). If $B = A$ then B is an improper subset of A. We define a partition $\mathcal{P} = \{P_i\}$ of a set A as a collection of disjoint sets such that $\cup_i P_i = A$ and $P_i \cap P_j = \emptyset$ for all $i \neq j$. We use the notation $A[k]$ to denote the state of set A at time k.

We start by defining the *ζ-variables* (read zeta variables). The efficient calculation of these variables is the basisfor the methods presented.

Definition 1. Consider a set A and let O_A and U_A denote the observed and unobserved sites in A. We define the *ζ-variables* (read zeta variables) for set A as

$$\zeta_{A,j}[k] = P(U_A[k] = j | O^{0,k}), \text{ where } 1 \leq j \leq n_A. \tag{4.1.4.1}$$

Then $\zeta_{A,j}[k]$ is the probability that the set of unobserved sites in A is in state j at time k, given the observations up to that time. In Equation 4.1.4.1, n_A is the number of states that U_A may take and is given by $n_A = \prod_{i \in U_A} m_i$.

Definition 2. Let $\mathcal{P} = \{A, B, \bar{I}\}$ be a partition of the network graph $\Gamma[\mathbf{D}']$ into an arbitrary set A, a set $\bar{I}$ containing all the non-influencing sites outside of A, and a set B. We denote by O_A (O_B) and U_A (U_B) the observed and unobserved sites in A (B). We refer to such a partition of the network as a *basic partition* for set A.

Definition 3. Consider a basic partition of an arbitrary set A. A is *ζ-independent* (read zeta-independent) if

(i) All the incoming arrows into A are from observed sites.

(ii) The only outgoing arrows from unobserved sites in A are to non-influencing sites.

Condition (i) just says that there can be no arrow from an unobserved site in B to A, or from $\bar{I}$ to A. Note that this means that $P(U_A[k+1]|S[k]) = P(U_A[k+1]|U_A[k], O^k)$, which can be easily computed by running the influence model forward one time step. Condition (ii) says that no site in U_A may influence a site in B. This means that all the incoming arrows into B are from observed sites, so that $P(U_B[k+1]|S[k]) = P(U_B[k+1]|U_B[k], O^k)$. Figure 4-2 illustrates this situation.

We will see shortly that there is an efficient algorithm to calculate the ζ-variables of a ζ-independent set. Before we go on, however, we make some remarks about ζ-independent sets that will come in handy shortly.

Corollary 4.1.1. *Let the set A be ζ-independent and consider its basic partition, as shown in Figure 4-2. Then the set B, which is formed by all sites that are not in A, except the non-influencing ones, is ζ-independent.*

Proof. The set A is ζ-independent so its basic partition looks like Figure 4-2. From the definition of ζ-independence, we know that all the arrows into A are from observed sites. Therefore all the outgoing arrows from unobserved sites in B must go to non-influencing sites. Similarly, since all the outgoing arrows from unobserved sites in A go to non-influencing sites, then all the incoming arrows into B must be from observed sites. Therefore B is ζ-independent. □

Corollary 4.1.2. *The union of two ζ-independent sets is also ζ-independent.*

Proof. Let two sites A_1 and A_2 be ζ-independent. Let $A = A_1 \cup A_2$ be the union of A_1 and A_2. Since all the incoming arrows into A_1 and A_2 are from observed sites, all the incoming arrows into A are from observed sites. Similarly, since all the outgoing arrows from unobserved sites in A_1 and A_2 are to non-inlfuencing sites, then all the outgoing arrows from unobserved sites in B are to non-influencing sites as well. Therefore A is ζ-independent. □

It turns out that if a set A is ζ-independent, then

$$P(A[k], B[k], U_A[k-1], U_B[k-1]|O^{0,k-1}) = P(A[k], U_A[k-1]|O^{0,k-1})P(B[k], U_B[k-1]|O^{0,k-1}),$$

where B is the set in the basic partition for A. This is stated in Theorem 4.1.4 below. However, before we present this theorem we need to prove a lemma.

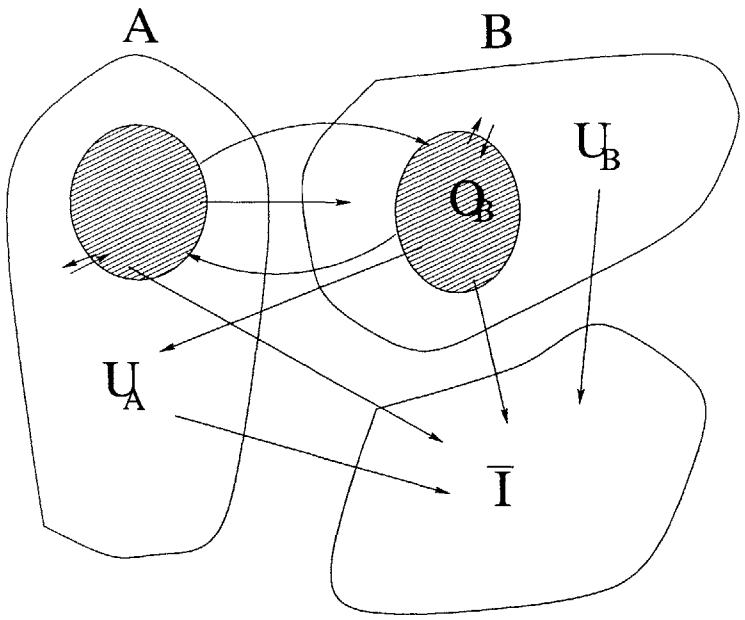

Figure 4-2: All the incoming arrows to set A are from observed sites, and all outgoing arrows from the unobserved sites of A are to non-influencing sites.

Lemma 4.1.3. *Consider a ζ-independent set A and its basic partition. Then the events that sets U_A and U_B are in state j_A and j_B, respectively, at time k given the observations up to that time are independent. That is,*

$$P\big(U_A[k] = j_A, U_B[k] = j_B|O^{0,k}\big) = P\big(U_A[k] = j_A|O^{0,k}\big)P\big(U_B[k] = j_B|O^{0,k}\big).$$

Proof. We prove this lemma by induction. Consider the basic partition for a ζ-independent

set A (see Figure 4-2). From the POIM definition, we have that for $k = 0$,

$$\begin{aligned} P(U_A[0] = j_A, U_B[0] = j_B|O^0) &= \Big(\prod_{i\in U_A} \pi_i[j_i]\Big)\Big(\prod_{i\in U_B} \pi_i[j_i]\Big) \\ &= P(U_A[0] = j_A|O^0)P(U_B[0] = j_B|O^0), \end{aligned}$$

where j_i denotes the status of site i that is consistent with any state of the network where $U_A = j_A$ and $U_B = j_B$.

Now we assume the claim is true for time k, and we show it is true for time $k+1$. The induction hypothesis is therefore that $P(U_A[k], U_B[k]|O^{0,k}) = P(U_A[k]|O^{0,k})P(U_B[k]|O^{0,k})$.

We then express the probability we are interested in as

$$P(U_A[k+1], U_B[k+1]\big|O^{0,k+1}) = \frac{P(U_A[k+1], U_B[k+1], O^{k+1}\big|O^{0,k})}{P(O^{k+1}\big|O^{0,k})}, \tag{4.1.4.2}$$

and calculate the numerator and the denominator separately. The numerator becomes

$$\begin{aligned} N &= \sum_{U_A[k]}\sum_{U_B[k]} P\big(U_A[k+1], U_B[k+1], O^{k+1}\big|U_A[k], U_B[k], O^{0,k}\big)P\big(U_A[k], U_B[k]\big|O^{0,k}\big) \\ &= \sum_{U_A[k]}\sum_{U_B[k]}\Big(P\big(U_A[k+1], O_A^{k+1}\big|U_A[k], O^{0,k}\big) \\ &\quad \times P\big(U_B[k+1], O_B^{k+1}\big|U_B[k], O^{0,k}\big)P\big(U_A[k]\big|O^{0,k}\big)P\big(U_B[k]\big|O^{0,k}\big)\Big) \\ &= \Big(\sum_{U_A[k]} P\big(U_A[k+1], O_A^{k+1}\big|U_A[k], O^k\big)P\big(U_A[k]\big|O^{0,k}\big)\Big) \\ &\quad \times\Big(\sum_{U_B[k]} P\big(U_B[k+1], O_B^{k+1}\big|U_B[k], O^k\big)P\big(U_B[k]\big|O^{0,k}\big)\Big) \\ &= N_1 N_2, \end{aligned}$$

where

$$N_1 = \sum_{U_A[k]} P\big(U_A[k+1], O_1^{k+1}\big|U_A[k], O^k\big)P\big(U_A[k]\big|O^{0,k}\big) = P\big(U_A[k+1], O_A^{k+1}\big|O^{0,k}\big), \tag{4.1.4.3}$$

and

$$N_2 = \sum_{U_B[k]} P\big(U_B[k+1], O_B^{k+1} \big| U_B[k], O^k\big) P\big(U_B[k] \big| O^{0,k}\big) = P\big(U_B[k+1], O_B^{k+1} \big| O^{0,k}\big). \tag{4.1.4.4}$$

The denominator becomes

$$\begin{aligned} P(O_{k+1}|O^{0,k}) &= \sum_{U_A[k]} \sum_{U_B[k]} P\big(O_A^{k+1}, O_B^{k+1} \big| U_A[k], U_B[k], O^{0,k}\big) P\big(U_A[k], U_B[k] \big| O^{0,k}\big) \\ &= \sum_{U_A[k]} \sum_{U_B[k]} P\big(O_A^{k+1} \big| U_A[k], O^k\big) P\big(O_B^{k+1} \big| U_B[k], O^{0,k}\big) P\big(U_A[k] \big| O^{0,k}\big) P\big(U_B[k] \big| O^{0,k}\big) \\ &= D_1 D_2, \end{aligned} \tag{4.1.4.5}$$

where

$$D_1 = \sum_{U_A[k]} P\big(O_A^{k+1} \big| U_A[k], O^k\big) P\big(U_A[k] \big| O^{0,k}\big) = P\big(O_A^{k+1} \big| O^{0,k}\big) \tag{4.1.4.6}$$

and

$$D_2 = \sum_{U_B[k]} P\big(O_B^{k+1} \big| U_B[k], O^k\big) P\big(U_B[k] \big| O^{0,k}\big) = P\big(U_B^{k+1} \big| O^{0,k}\big). \tag{4.1.4.7}$$

Putting these together we find that

$$\begin{aligned} P(U_A[k+1], U_B[k+1] \big| O^{0,k+1}) &= \frac{N_1 N_2}{D_1 D_2} \\ &= \frac{P\big(U_A[k+1], O_A^{k+1} \big| O^{0,k}\big) P\big(U_B[k+1], O_B^{k+1} \big| O^{0,k}\big)}{P\big(O_A^{k+1} \big| O^{0,k}\big) P\big(O_B^{k+1} \big| O^{0,k}\big)} \\ &= \left(\frac{P\big(U_A[k+1], O_A^{k+1} \big| O^{0,k}\big)}{P\big(O_A^{k+1} \big| O^{0,k}\big)}\right) \left(\frac{P\big(U_B[k+1], O_B^{k+1} \big| O^{0,k}\big)}{P\big(O_B^{k+1} \big| O^{0,k}\big)}\right) \\ &= P\big(U_A[k+1] \big| O^{0,k}, O_A^{k+1}\big) P\big(U_B[k+1] \big| O^{0,k}, O_B^{k+1}\big). \end{aligned}$$

To finish the proof, we need to show that $P\big(U_A[k+1] \big| O^{0,k}, O_A^{k+1}\big) = P\big(U_A[k+1] \big| O^{0,k+1}\big)$ and that $P\big(U_B[k+1] \big| O^{0,k}, O_B^{k+1}\big) = P\big(U_B[k+1] \big| O^{0,k+1}\big)$. We do this for U_A next. We start

with

$$P(U_A[k+1]|O^{0,k+1}) = \frac{P(U_A[k+1], O_{k+1}|O^{0,k})}{P(O_{k+1}|O^{0,k})}, \tag{4.1.4.8}$$

and calculate the numerator and denominator separately. The numerator becomes

$$\begin{aligned}
P(U_A[k+1], O^{k+1}|O^{0,k}) &= \sum_{U_A[k]} \sum_{U_B[k]} P\big(U_A[k+1], O^{k+1}\big|U_A[k], U_B[k], O^{0,k}\big) P\big(U_A[k], U_B[k]\big|O^{0,k}\big) \\
&= \sum_{U_A[k]} \sum_{U_B[k]} \Big(P\big(U_A[k+1], O_A^{k+1}\big|U_A[k], O^{0,k}\big) P\big(O_B^{k+1}\big|U_B[k], O^{0,k}\big) \\
&\quad \times P\big(U_A[k]\big|O^{0,k}\big) P\big(U_B[k]\big|O^{0,k}\big) \Big) \\
&= \Big(\sum_{U_A[k]} P\big(U_A[k+1], O_A^{k+1}\big|U_A[k], O^k\big) P\big(U_A[k]\big|O^{0,k}\big) \Big) \\
&\quad \times \Big(\sum_{U_B[k]} P\big(U_B^{k+1}\big|U_B[k], O^k\big) P\big(U_B[k]\big|O^{0,k}\big) \Big) \\
&= P\big(U_A[k+1], O_A^{k+1}\big|O^{0,k}\big) P\big(O_B^{k+1}\big|O^{0,k}\big)
\end{aligned}$$

From Equation 4.1.4.5 we know that the denominator is to be

$$P(O^{k+1}|O^{0,k}) = P\big(O_A^{k+1}|O^{0,k}\big) P\big(O_B^{k+1}|O^{0,k}\big).$$

Finally, substituting into Equation 4.1.4.8 we see that

$$\begin{aligned}
P(U_A[k+1]|O^{0,k+1}) &= \frac{P\big(U_A[k+1], O_A^{k+1}|O^{0,k}\big) P\big(O_B^{k+1}|O^{0,k}\big)}{P\big(O_A^{k+1}|O^{0,k}\big) P\big(O_B^{k+1}|O^{0,k}\big)} \\
&= \frac{P\big(U_A[k+1], O_A^{k+1}|O^{0,k}\big)}{P\big(O_A^{k+1}|O^{0,k}\big)} \\
&= P\big(U_A[k+1]|O^{0,k}, O_A^{k+1}\big).
\end{aligned} \tag{4.1.4.9}$$

The proof that $P\big(U_B[k+1]|O^{0,k}, O_B^{k+1}\big) = P\big(U_B[k+1]|O^{0,k+1}\big)$ is exactly the same as this one, if we just interchange the A and B in all the previous equations, so we omit it here. This completes the proof. □

Theorem 4.1.4. *Consider a ζ-independent A and its basic partition. Then*

$$P(A[k], B[k], U_A[k-1], U_B[k-1]|O^{0,k-1}) = P(A[k], U_A[k-1]|O^{0,k-1})P(B[k], U_B[k-1]|O^{0,k-1}). \tag{4.1.4.10}$$

That is, if the set A is *ζ-independent* then the probability that the unobserved sites in A take on a given state at time $k-1$ and that set A takes a given state at time k, conditioned on the observations up to time $k-1$, is independent of the state of set B at time k, and of the state of the unobserved sites in B at $k-1$ for all times $k \geq 1$.

Proof. Consider a ζ-independent set A and its basic partition (see Figure 4-2). We want to show that $P(A[k+1], B[k+1], U_A[k], U_B[k]|O^{0,k}) = P(A[k+1], U_A[k]|O^{0,k})P(B[k+1], U_B[k]|O^{0,k})$ for all times greater than 0. We do this by induction.

$$\begin{aligned}
P(A[k+1], B[k+1], U_A[k], U_B[k]|O^{0,k}) &= P(A[k+1], B[k+1]|U_A[k], U_B[k], O^{0,k}) \\
&\quad \times P(U_A[k], U_B[k]|O^{0,k}) \\
&= P(A[k+1], B[k+1]|U_A[k], U_B[k], O^{0,k}) \\
&\quad \times P(U_A[k]|O^{0,k})P(U_B[k]|O^{0,k}) \\
&= P(A[k+1]|U_A[k], O^{0,k})P(B[k+1]|U_B[k], O^{0,k}) \\
&\quad \times P(U_A[k]|O^{0,k})P(U_B[k]|O^{0,k}) \\
&= P(A[k+1]|U_A[k], O^{0,k})P(U_A[k]|O^{0,k}) \\
&\quad \times P(B[k+1]|U_B[k], O^{0,k})P(U_B[k]|O^{0,k}) \\
&= P(A[k+1], U_A[k]|O^{0,k})P(B[k+1], U_B[k]|O^{0,k}).
\end{aligned}$$

□

Remark 1. Consider a ζ-independent set A and its basic partition. Then, for $k \geq 1$, the

following relations hold:

$$\begin{aligned}
P(U_A[k-1], U_B[k-1]|O^{0,k-1}) &= P(U_A[k-1]|O^{0,k-1}) \\
&\quad \times P(U_B[k-1]|O^{0,k-1}) && (4.1.4.11) \\
P(O_A[k], O_B[k]|O^{0,k-1}) &= P(O_A[k]|O^{0,k-1})P(O_B[k]|O^{0,k-1}) && (4.1.4.12) \\
P(U_A[k], O_A^k, U_B[k], O_B^k|O^{0,k-1}) &= P(U_A[k], O_A^k|O^{0,k-1}) \\
&\quad \times P(O_B^k, U_B[k]|O^{0,k-1}) && (4.1.4.13) \\
P(U_A[k], U_B[k]|O^{0,k}) &= P(U_A[k]|O^{0,k})P(U_B[k]|O^{0,k}) \\
&= P(U_A[k]|O^{0,k-1}, O_A^k) \\
&\quad \times P(U_B[k]|O^{0,k-1}, O_B^k). && (4.1.4.14)
\end{aligned}$$

Equation 4.1.4.11 follows from adding over $A[k]$ and $B[k]$ in Equation 4.1.4.10. Equation 4.1.4.12 follows from adding over the unobserved sites in both A and B at times k and $k-1$ in Equation 4.1.4.10. Equation 4.1.4.13 follows from adding over the unobserved sites in A and B at time $k-1$ in Equation 4.1.4.10. Finally, Equation 4.1.4.14 follows from dividing Equation 4.1.4.13 over Equation 4.1.4.12. These four equations will be used throughout the rest of these section.

We introduced the concept of a ζ-independent set because we claimed that the ζ-variables of such a set were easily computed. In the next corollary, we specify how to compute them.

Corollary 4.1.5. *Consider a set A and its basic partition, and let A be ζ-independent. Then:*

(i) The following probabilities

(a) $p_{ij}^A[k] = P\big(U_A[k+1] = j, O_A^{k+1}|U_A[k] = i, O^k\big)$ and

(b) $p_i^{O_A}[k] = P\big(O_A^{k+1}|U_A[k] = i, O^k\big)$,

may be calculated by running the influence model forward one time step, for all i, j and $k \geq 0$.

(ii) The associated ζ-variables, $\zeta_{A,j}[k] = P(U_A[k] = j|O^{0,k})$, may be calculated by the following algorithm:

1) Initialization:

$$\zeta_{A,j}[0] = \prod_{i \in U_A} \pi_i[j_i] \qquad \forall j, \tag{4.1.4.15}$$

where j_i is the status of site i that is consistent with the state j of U_A.

2) Recursion:

$$\zeta_{A,j}[k+1] \;=\; \frac{\sum_{i=1}^{n_A} p_{ij}^{A}[k]\zeta_{A,i}[k]}{\sum_{i=1}^{n_A} p_{i}^{O_A}[k]\zeta_{A,i}[k]}, \tag{4.1.4.16}$$

for all j and for all $k \geq 0$, where n_A is the number of states that the set U_A may take.

Proof. Since A is ζ-independent, any site in A is influenced by other sites in A and/or observed sites outside of A. Therefore, conditioning on the state of set A and of those observed sites outside of A at time k, we may calculate the probability that any subset of A is in any given state at time k by running the influence model forward one time step. This proves (i) above.

The initialization of the algorithm follows from the definition of an IM, and the fact that $\zeta_{A,j}[0] = P(U_A[0] = j)$. Now we show why the recursion in Equation 4.1.4.16 holds:

$$\begin{aligned}
\zeta_{A,j}[k+1] &= P(U_A[k+1] = j | O^{0,k+1}) \\
&= P(U_A[k+1] = j | O^{0,k}, O_A^{k+1}) \qquad \text{by 4.1.4.14} \\
&= \frac{P(U_A[k+1] = j, O_A^{k+1} | O^{0,k})}{P(O_A^{k+1} | O^{0,k})} \\
&= \frac{\sum_{i=1}^{n_A} P(U_A[k+1] = j, O_A^{k+1}, U_A[k] = i | O^{0,k})}{\sum_{i=1}^{n_A} P(O_A^{k+1}, U_A[k] = i | O^{0,k})} \\
&= \frac{\sum_{i=1}^{n_A} P(U_A[k+1] = j, O_A^{k+1} | U_A[k] = i, O^{0,k}) P(U_A[k] = i | O^{0,k})}{\sum_{i=1}^{n_A} P(O_A^{k+1} | U_A[k] = i, O^{0,k}) P(U_A[k] = i | O^{0,k})} \\
&= \frac{\sum_{i=1}^{n_A} p_{ij}^{A}[k]\zeta_{A,i}[k]}{\sum_{i=1}^{n_A} p_{i}^{O_A}[k]\zeta_{A,i}[k]}.
\end{aligned}$$

□

The algorithm above explains why we are interested in ζ-independent sets: one may find the probability distributions of any ζ-independent set at time k, given the observations up to that time independently of the rest of the network, as described by Corollary 4.1.5. From earlier discussions in Chapter 3 we know that running the influence model forward one step takes on the order of $n^2m + m^2n$ computations for the homogeneous IM, so that evaluating each $p_{ij}^{A}[k]$ and $p_{i}^{O_A}[k]$ takes on the order of n^2m+m^2n. Therefore using the algorithm above to find the ζ-variables for T time steps of a ζ-independent set that may take n_A possible states in a homogeneous POIM, takes on the order of $Tn_A^2(n^2m + m^2n)$.

The concept of ζ-independence will allow us to calculate the probability of an observation sequence $P(O^{0,k})$ in a much more efficient way than that presented in Chapter 3, as well as to do some estimation both for ζ-independent sets and for the whole network. Before we proceed, however, we need to introduce the concepts of *ζ-minimal sets* and of a *ζ-minimal partition* of the network graph $\Gamma[\mathbf{D}']$.

Let s_i be site i of the network graph $\Gamma[\mathbf{D}']$ in an IM. Assume we want to find the smallest ζ-independent set that contains site i. From its definition, a ζ-independent set that contains site s_i must also contain all the influencing sites that s_i directly influences, as well as any unobserved site that influences s_i directly. We may call this set S_1. Now, if there is any hope for S_1 to be ζ-independent, for each site in it, all the influencing and observed sites it influences, and all the unobserved sites that influence it should be in S_1 so that ζ-independence is assured. Therefore we may think of a set S_2 that contains all such sites. We may keep doing this until we get a set S that is ζ-independent. This set would be the smallest ζ-independent set (in number of sites) that contains site s_i. This situation is captured by the following definitions.

Definition 4. Let the *generating function* $\sigma : S_1 \rightarrow S_2$ be a function that takes as argument a set S_1 and maps it to a set S_2 such that for each site s_i in S_1, S_2 contains all the unobserved sites that influence s_i directly, and, if the s_i is unobserved, all the observed and influencing sites it directly influences.

Under this definition, any set S that satisfies $\sigma(S) = S$ is ζ-independent. Similarly,

we may start with any arbitrary non-empty set of sites and apply the generating function recursively to it until we get a ζ-independent set. Because there are n sites in the network, this would take at most n iterations, so that $\sigma^n(S)$ is ζ-independent for any set S.

Definition 5. We say that a non-empty set (or site) S_1 *generates* the set $S_2 =< S_1 >$ if $S_2 = \sigma^n(S_1)$. We also say that S_1 is a *generator* of S_2.

At every iteration of the function σ, the resulting set is made from the original one plus sites that violated the conditions for ζ-independence. The set $S_2 = \sigma^n(S_1)$ is thus the smallest set that contains S_1 and that is ζ-independent.

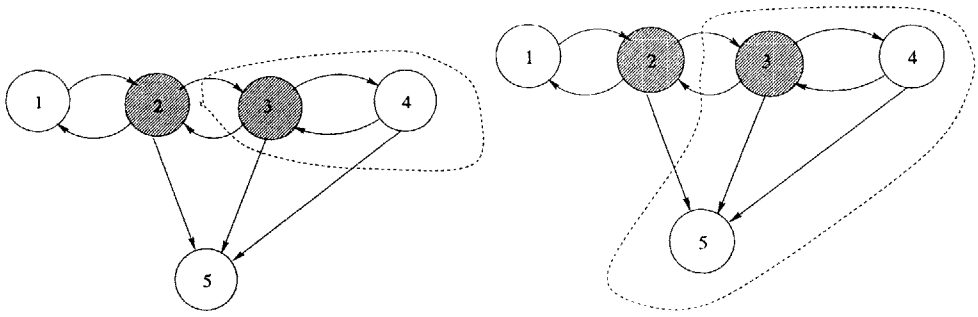

Figure 4-3: This figure illustrates how the generating function σ acts on sets. In the left figure, we see that applying the generating function to site 3 or 4 yields the set $\{3,4\}$: $\sigma(\{3\}) = \sigma(\{4\}) = \{3,4\}$. On the figure on the right, we consider the set generated by $\sigma^2(\{5\})$. Applying σ to site 5 yields the set $\{4,5\}$. Applying σ to this set once yields the set inside the dotted line. we may say, for example, that site 5 generates the set $\{3,4,5\}$.

Definition 6. Let A be a set of sites in the network graph $\Gamma[\mathbf{D}']$. A is a ζ-*minimal set* if

(i) it is ζ-independent, and

(ii) no other set that contains a subset of A is ζ-independent.

Corollary 4.1.6. *Every site in a ζ-minimal set is a generator of the set.*

Proof. Let A be a ζ-minimal set and let i be a site in A. We know that $< i >$ is the smallest ζ-independent set that includes site i, so that $< i >$ must be a subset of A. However, since no subset of A may be ζ-independent, it follows that $< i >= A$. □

Theorem 4.1.7. *A set A is ζ-minimal if and only if there exists an influencing or observed site that generates it.*

Proof. Assume A is ζ-minimal. Then, it must not include any non-influencing site. This follows from the fact that if we have a ζ-independent set and remove all the non-influencing sites in it, the resulting set is also ζ-independent. Then, Corollary 4.1.6 tells us that any site in it is a generator, which proves the only if case. Now, assume that an observed or influencing site i generates A, so that $< i >= A$. Then A is certainly ζ-independent. To prove that no subset of A is ζ-independent, assume there exists a proper subset B of set A that is ζ-independent. If $i \in B$, then because $< i >$ is the smallest ζ-independent set that contains site i, B cannot be ζ-independent, which is a contradiction. Now, assume i is not in B and let j be any site in B. Because $A =< i >$, there exists an integer $n_o \leq n$ such that j is not in $\sigma^{n_o}(i)$ but $j \in \sigma^{n_o+1}(i)$. Then, there must be an arrow from an unobserved site in $\sigma^{n_o}(i)$ to j, or from j into $\sigma^{n_o}(i)$, either of which make A not ζ-independent, which is a contradiction. □

Definition 7. Consider a partition of the influencing and observed sites of the network into the largest number of ζ-minimal sets and call them A_1,..., A_{n_m}. Then the partition of the network $\mathcal{P} = \{\bar{I}, A_1, ..., A_{n_m}\}$, where $\bar{I}$ is the set of all non-influencing sites is a *ζ-minimal partition* of the network.

Theorem 4.1.8. *The ζ-minimal partition for a network graph $\Gamma[\mathbf{D}']$ is unique.*

Proof. Assume there exist two different ζ-minimal partitions for a network graph $\Gamma[\mathbf{D}']$, and denote them by $\mathcal{P}_1 = \{\bar{I}, A_1, ..., A_{n_m}\}$ and $\mathcal{P}_2 = \{\bar{I}, B_1, ..., B_{n_m}\}$. Let i be a site in A_i and let B_i be the set of partition $\mathcal{P}_2$ that contains i. Since A_i is ζ-minimal, then $A_i =< i >$, which is the smallest ζ-independent set that contains i. Since B_i is ζ-independent then A_i must be a subset of B_i. Furthermore, since B_i is ζ-minimal there is no proper subset of it that is ζ-independent so that that $B_i = A_i$. □

Now we propose a way to find the minimal partition of a network graph $\Gamma[\mathbf{D}']$. We do so by creating a modified graph $\Gamma[\mathbf{D}'_{\mathbf{m}}]$ as described by the following theorem.

Theorem 4.1.9. *Let $\overline{I}$ and O be the non-influencing and observed sites of the network graph $\Gamma[\mathbf{D}']$. Starting with the graph $\Gamma[\mathbf{D}']$ create the modified network graph $\Gamma[\mathbf{D}'_{\mathbf{m}}]$ by following this two steps, in any order:*

1. *Remove all the non-influencing sites $\overline{I}$, and the arrows to them*

2. *Remove all the arrows from the observed sites O.*

Let n_m be the number of disconnected components of the modified graph $\Gamma[\mathbf{D}'_{\mathbf{m}}]$ and let A_i be the ith such disconnected component. Define a partition $\mathcal{P}$ of the original network graph $\Gamma[\mathbf{D}']$ by $\mathcal{P} = \{\overline{I}, A_1, ..., A_{n_m}\}$. The partition $\mathcal{P}$ is the ζ-minimal partition of the original graph $\Gamma[\mathbf{D}']$.

Proof. Consider a set A that is a disconnected component of the modified graph $\Gamma[\mathbf{D}'_{\mathbf{m}}]$, as constructed above. We need to show that A is ζ-minimal. Because of the way $\Gamma[\mathbf{D}'_{\mathbf{m}}]$ was created, A has no non-influencing sites. The only incoming arrows to A in the original graph $\Gamma[\mathbf{D}']$ could have then been from observed sites. Similarly, all the outgoing arrows from unobserved sites in A in the original graph $\Gamma[\mathbf{D}']$ must have been to non-influencing sites. Then A is ζ-independent. Now consider a proper subset of A, and call it B. Then B is not a disconnected component of $\Gamma[\mathbf{D}'_{\mathbf{m}}]$, which means that in the original graph $\Gamma[\mathbf{D}']$, it must have either an incoming arrow from an unobserved influencing site, or an outgoing arrow from an unobserved site to an observed or influencing site. Then B is not ζ-independent. Thus, A is ζ-minimal. Because all the sets in the partition, except $\overline{I}$ are ζ-minimal, it is impossible to split any of them and obtain a ζ-minimal set which makes this partition the one with the largest number of ζ-minimal sets. □

The ζ-minimal partition will be yield an efficient method to estimate the state of the influencing sites of the network, and the probability of an observation sequence in a POIM. However, it will not be useful if we want to estimate the state of the whole network, including the non-influencing sites. To deal with this situation, we need to work with a different partition of the network, the *ζ-network partition*. To describe this partition of the network, we must define a different generating function that incorporates non-influencing sites into ζ-independent sets.

Definition 8. A partition of the network $\Gamma[\mathbf{D}']$ into the largest number of ζ-independent sets is a *ζ-network partition.*

We say that site i directly influences site j is there is an arrow in the network graph from i into j. We say that site i influences site j indirectly if there is at least one path from site i to site j that does not go through an observed site.

Definition 9. A ζ-independent set A is *η-minimal* if there is no unobserved site in it that directly or indirectly influences an unobserved site outside it, and if it cannot be split up into ζ-independent sets.

If a set is ζ-independent, then the only outgoing arrows from unobserved sites in it may be to non-influencing site. An η-minimal set may not have such arrows. In a ζ-network partition every site in the network must be included in a ζ-independent set. This implies that each non-influencing site needs to belong to a ζ-independent set, so that there will not be any arrow from an unobserved site in any of these sets into a non-influencing site outside it. Also, since this partition has by definition the largest number of disjoint ζ-independent sets that cover the network graph, then none of these ζ-independent sets can be divided into two or more ζ-independent sets. Therefore each of the ζ-independent sets that define a ζ-network partition must be η-minimal.

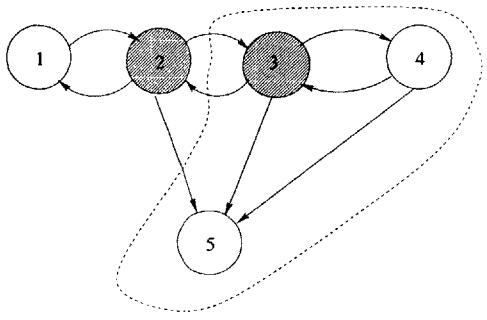

Figure 4-4: This figure illustrates how the generating function η acts on sets. Applying the generating function η once to site 4 yields the set $\{3,4,5\}$: $\eta(\{4\}) = \{3,4,5\}$. Similarly, $\eta(\{3\}) = \{3,4\}$ and $\eta^2(\{3\}) = \{3,4,5\}$.

Definition 10. Let the *network generating function* $\eta : S_1 \rightarrow S_2$ be a function that takes as argument a set S_1 and maps it to a set S_2 such that for each site s_i in S_1, S_2 contains all

the unobserved sites that influence s_i directly, and, if the site s_i is unobserved, all the sites it directly influences.

Definition 11. We say that a non-empty set (or site) S_1 *η-generates* the set $S_2 = [S_1]$ if $S_2 = \eta^n(S_1)$. We also say that S_1 is an *η-generator* of S_2.

Figure 4-4 illustrates these definitions. Any set S that satisfies $\eta(S) = S$ is ζ-independent (all the arrows into and out of S are from observed sites). We may start with any arbitrary non-empty set of sites and apply the network generating function recursively to it until it converges. Because there are n sites in the network, this would take at most n iterations, so that $\eta^n(S)$ is ζ-independent for any set S.

Assume we iteratively apply the generating function i times to an set S_o so that $S_i = \eta^i(S_o) = \eta(S_{i-1})$. Then at each iteration k, the set S_k is made from the sites in S_{k-1} plus sites that either violated the conditions for ζ-independence, or that are non-influencing and have an incoming arrow from an unobserved site S_{k-1}. Therefore if a set is to be η-minimal and contain S_{k-1} it must include S_k. This is true at every k. Since we know that $\eta^k(S_o)$ converges for in at most $n-1$ steps, then the set $S = \eta^n(S_o)$ is the smallest set that contains S_o, is ζ-independent, and contains all non-influencing sites that the unobserved sites in it may influence directly or indirectly (it has no outgoing arrows from unobserved sites). Thus it is η-minimal.

Corollary 4.1.10. *Every site in an η-minimal set η-generates it.*

Proof. Let i be a site of the η-minimal set A. Let $[i]$ be a subset of A. From our reasoning in the preceding paragraph we know that $[i]$ is η-minimal. Then consider the set B of all sites in A that are not in $[i]$. Because of the way $[i]$ is generated from the function η, we know that the only arrows into it must be from observed sites and that all the arrows from it are from observed sites as well. Then all the arrows into B from $[i]$ are from observed sites and all the arrows from B into $[i]$ are from observed sites. Since A is ζ-independent, then we know that all other arrows from and into B satisfy ζ-independence. Thus B is ζ-independent. Then A can be split up into two ζ-independent sets, which is a contradiction since A is η-minimal. □

Theorem 4.1.11. *A set A is η-minimal if and only if there exists a site $i \in A$ such that $A = [i]$.*

Proof. Assume A is η-minimal, and let i be a site in it. Then by Corollary 4.1.10 any site in it η-generates it. Conversely, assume a set A is such that it is η-generated by a single site: $A = [i]$ for some i. Then$A$ is the smallest ζ-independent set that contains i and that has no outgoing arrows from unobserved sites, as explained in the previous remark. Assume it may be split into two ζ-independent sets B and C and let $i \in B$. Then B must have an unobserved site with an outgoing arrow to site j in C. Then there is an incoming arrow from an unobserved site outside of C into C which violates ζ-independence of C. This is a contradiction since C is ζ-independent by assumption. □

Theorem 4.1.12. *The ζ-network partition of an influence graph is unique.*

Proof. Assume there exist two different ζ-network partitions for a the graph $\Gamma[\mathbf{D}']$, and denote them by $\mathcal{P}_1 = \{A_1, ..., A_{n_n}\}$ and $\mathcal{P}_2 = \{B_1, ..., B_{n_n}\}$. Then all sets A_i and B_i must be η-minimal. Let i be a site in A_i and let B_i be the set of partition $\mathcal{P}_2$ that contains i. Since A_i is η-minimal, then $A_i =< i >$ by Theorem 4.1.11 and Corollary 4.1.10. Since $< i >$ is the smallest ζ-independent set that contains i and has no outgoing arrows from unobserved sites, then A_i must be a subset of B_i. Interchanging A_i for B_i and vice-versa in the previous sentence, we find that B_i must be as subset of A_i. Since A_i and B_i are subsets of each other, they must be equal. Because the site i is arbitrary, it follows that the two partitions are equal. □

Now we propose a method to find the ζ-network partition of an arbitrary network graph $\Gamma[\mathbf{D}']$.

Theorem 4.1.13. *Let $\overline{I}$ and O be the non-influencing and observed sites of the network graph $\Gamma[\mathbf{D}']$. Starting with the graph $\Gamma[\mathbf{D}']$ create the modified network graph $\Gamma[\mathbf{D}'_\mathbf{n}]$ by removing all the arrows from the observed sites O.*

Let n_n be the number of disconnected components of the modified graph $\Gamma[\mathbf{D}'_\mathbf{n}]$ and let A_i be the ith such disconnected component. Define a partition $\mathcal{P}$ of the original network graph $\Gamma[\mathbf{D}']$ by $\mathcal{P} = \{A_1, ..., A_{n_n}\}$. The partition $\mathcal{P}$ is the ζ-network partition of the original graph $\Gamma[\mathbf{D}']$.

Proof. Consider a disconnected component A of the modified graph $\Gamma[\mathbf{D}'_{\mathbf{n}}]$, constructed as described above. We want to show that A is η-minimal, because then no set in the partition can be split into ζ-independent sets. Because of the way $\Gamma[\mathbf{D}'_{\mathbf{n}}]$ was constructed, then A has no outgoing or incoming arrows from unobserved sites, so that A is ζ-independent. Assume that we may split A into two ζ-independent sets B and C. Because no proper subset of A is a disconnected component of $\Gamma[\mathbf{D}'_{\mathbf{n}}]$, then there must be an arrow from an unobserved site in B (without loss of generality) into C. But this implies C is not η-independent, which is a contradiction. □

4.1.1 Some Examples

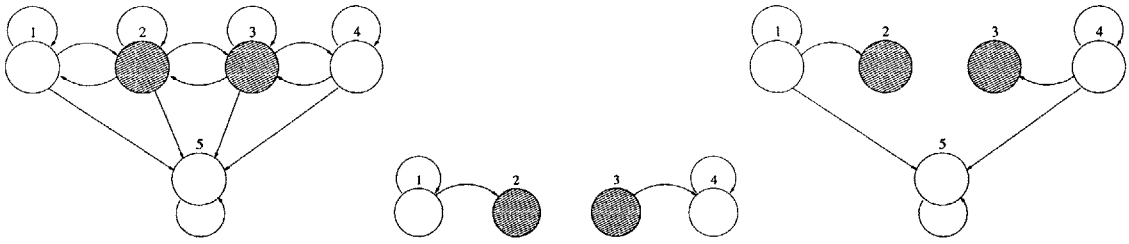

Figure 4-5: The figure on the left is the network graph of a POIM which ζ-minimal partition we wish to find. The figure on the middle is the modified graph resulting from erasing the non-influencing sites, the arrows to them, and the arrows from observed sites. Each disconnected component of this modified graph is a ζ-minimal set of the original graph. The figure on the right is the modified graph resulting from erasing the arrows from observed sites. Each disconnected component of this graph is a set of the ζ-network partition.

We present some examples that illustrate how to find the ζ-minimal and ζ-network partition of an influence graph, and some of the concepts introduced in this section.

Consider the network graph shown on the left side of Figure 4-5. Site 5 is the only non-influencing site of the graph. To create the ζ-minimal partition of the graph we follow the method suggested by Theorem 4.1.9: we create a modified network graph by removing the non-influencing sites (which is site 5), and all the arrows from the observed sites. The resulting network graph is shown on the right part of Figure 4-5. Theorem 4.1.9 tells us that

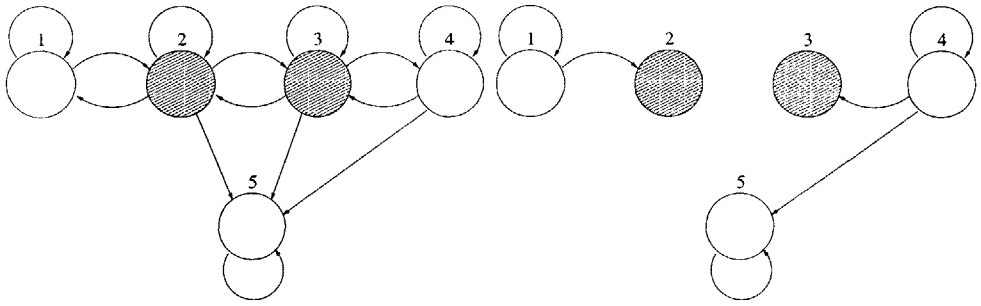

Figure 4-6: The figure on the left is the network graph of a POIM similar to the one in Figure 4-5. The ζ-minimal partition of the two network graphs is the same. However, they have different ζ-network partitions. The ζ-network partition for this network graph is shown on the right part of the figure.

the ζ-minimal sets of the ζ-minimal partition of the original graph are the disconnected sets of the modified graph. We therefore have that the ζ-minimal partition of this graph consists of two such sets: $A_1 = \{1, 2\}$ and $A_2 = \{3, 4\}$. The ζ-network partition is created by erasing the arrows from sites the observed sites 3 and 4. This modified graph is shown on the right of the figure. There is only one disconnected component, so the ζ-network partition consists of a single set, the whole network.

Consider now the network graph shown in Figure 4-6. This network graph is the same as that on the left of Figure 4-5 except that the arrow from site 1 to site 5 is missing. The two network graphs have the same ζ-minimal partitions. However, their ζ-network partition is very different. The one for the previous network graph consists of a single set of sites, as shown on the right of Figure 4-5. The one for this network graph consists of two sites: $A_1 = \{1, 2\}$ and $A_2 = \{3, 4, 5\}$. These two sets cover the whole graph and are ζ-independent. This example shows how similar network graphs might have the same ζ-minimal partitions, but different ζ-network partitions. As will be discussed later in the chapter, the more sets these partitions have, the easier computationally it is to do estimation on different parts of the network.

Now consider the network graph shown on the left of Figure 4-7. Site 9 is the only observed site and sites 2, 4, 6, and 8 are non-influencing. We create the modified graph shown on

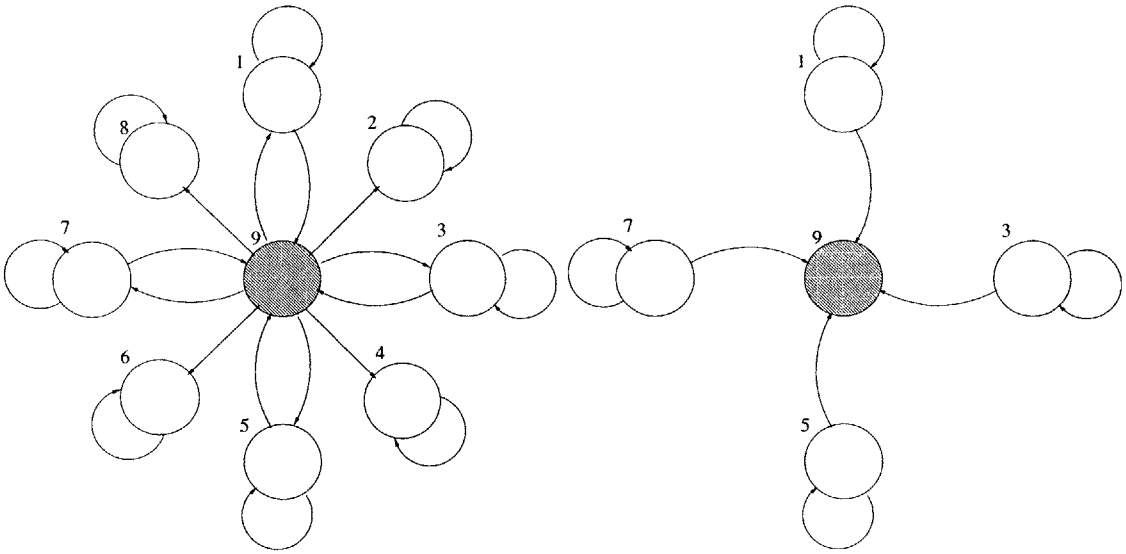

Figure 4-7: The figure on the left is the network graph of a POIM which ζ-minimal partition we wish to find. The figure on the right is the modified graph resulting from erasing the non-influencing sites, the arrows to them, and the arrows from observed sites. Each disconnected component of this modified graph is a ζ-minimal set of the original graph.

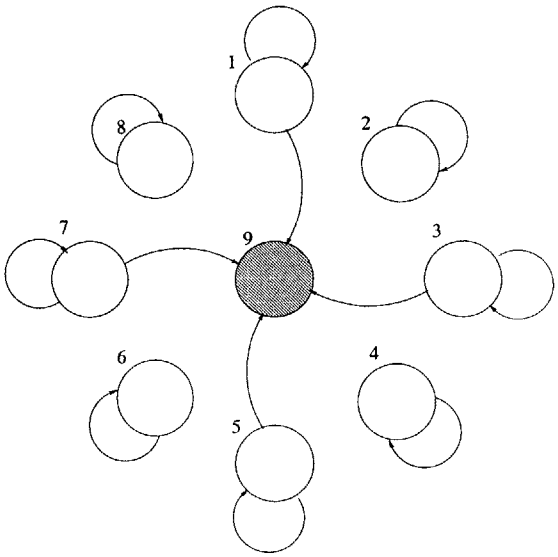

Figure 4-8: This figure is the modified graph to find the ζ-network partition of the network graph shown in Figure 4-7 above. It was created by removing the arrows from the observed sites in the original graph. Each disconnected component is ζ-independent and is a set of the ζ-network partition.

the middle of the same figure to find the ζ-minimal sets of the ζ-minimal partition. To create the modified graph, we erase the non-influencing sites 2, 4, 6, and 8, all the arrows to them, and all the arrows from observed sites. The ζ-minimal partition therefore has a single ζ-minimal set $A = \{1, 3, 5, 7, 9\}$. The ζ-network partition of this network graph, on the other hand, consists of the disconnected components of the graph shown in Figure 4-8. This graph was created by erasing the arrows from the observed sites in the original graph. Each disconnected component is ζ-independent. Therefore, the ζ-network partition of this network graph consists of the five sets $A_1 = \{1, 3, 5, 7, 9\}$, $A_2 = \{2\}$, $A_3 = \{4\}$, $A_4 = \{6\}$, and $A_5 = \{8\}$. We will see later in this chapter that the larger the number of sets that form the ζ-network partition, the easier the computation to estimate the state of the network will be.

Finally, consider the network graph shown in Figure 4-9. This network graph is very similar to that on the left of Figure 4-7. The ζ-minimal partition of both network graphs is in fact the same, as can be verified by creating the appropriate modified graph. The modified graphs used to find the ζ-network partitions of both network graphs are very different however. The one for the circular network graph on Figure 4-7 is shown on Figure 4-8 and consists of five disconnected sets. The one for this network graph is shown on the right of Figure 4-9 and consists of a single set.

Summarizing, we have defined the concepts of ζ-independent sets whose ζ-variables are computationally easy to find. Then we introducen ζ-minimal sets, and the ζ-minimal and ζ-network partitions of the network graph. Theorem 4.1.8 says there is a unique decomposition of the network graph into ζ-minimal sets, and Theorem 4.1.9 suggests a method to find this decomposition. These theorems along with the corollaries developed in this section provide the basis to find the probability of an observation sequence, and to do estimation on a POIM in a much more efficient way than that presented in Chapter 3. We develop these methods next.

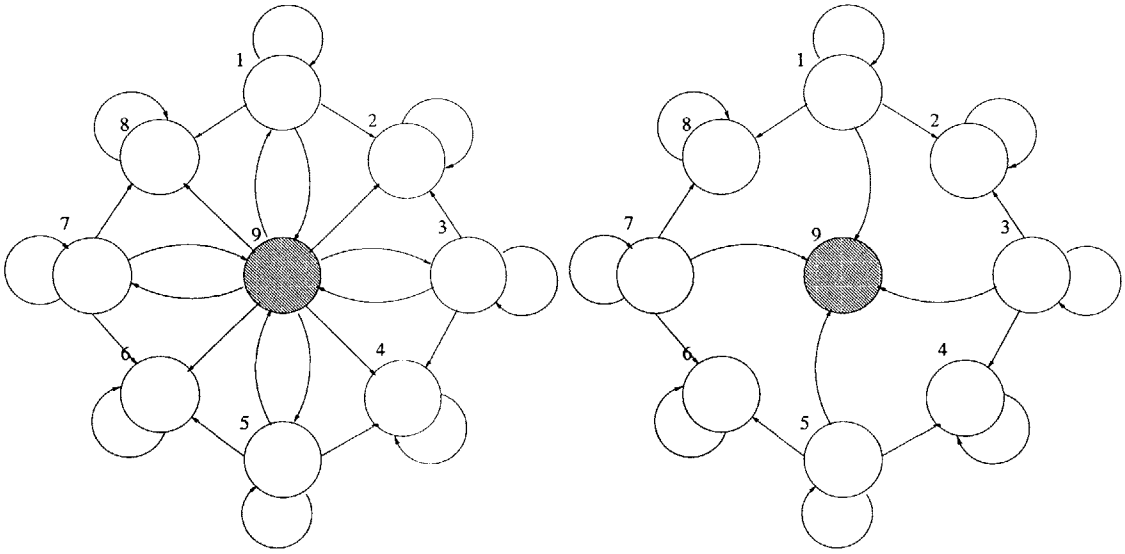

Figure 4-9: The figure on the left is the network graph of a POIM similar to that on the left of Figure 4-7. The modified graph resulting from erasing the arrows from observed sites is shown on the right. The disconnected components of this modified graph define the ζ-network partition of the network graph, which consists of a single set.

4.2 Online Algorithm to Estimate the Present State of Sets of Sites in The Network

Throughout this section, we address the following problem: Assume we wish to estimate the most likely state of the unobserved sites of a set of sites A at the present time given the observations up to the present. If we let $U_A^*[k]$ be the estimate at time k, then we may define this estimate as

$$U_A^*[k] = \arg \max_{1 \leq i \leq n_A} P(U_A[k] = i | O^{0,k}),$$

where n_A is the number of states that U_A may take: $n_A = \prod_{i \in U_A} m_i$. This problem could be solved by finding the forward variables for the whole network, as discussed in Section 3.1.1, and then adding over all the sites that are not in U_A to find the distribution $P(U_A[k] = i, O^{0,k})$. This would take on the order of TN^2n^2 computations, where $N = \prod_{i \in U} m_i$ is the number of possible states of the unobserved sites in the network. Therefore the running time of the algorithm is very large since N can become very large very soon. For example, if the number of unobserved sites is $n - 2$ and each site is known to be in one of two states, then the running time will be on the order of $Tn^2 2^{2n}$ computations.

It turns out that we can do much better by using the concepts introduced in Section 4.1.

4.2.1 Online Estimation for a ζ-Independent Set

If the set A which state we wish to estimate is ζ-independent then we may do the estimation by finding the zeta variables for that set at each time, based on the zeta variables at the previous time and the new observation, as described by Equations 4.1.4.15 and 4.1.4.16 . Then at time k we would set our estimate to be

$$U_A^*[k] = \arg \max_{1 \leq i \leq n_A} \zeta_{A,i}[k],$$

where n_A is the number of states that U_A may take: $n_A = \prod_{i \in U_A} m_i$. There are n_A ζ-variables at each time, and each will take on the order of $n^2 n_A$ computations, so that using this algorithm for T time steps will result in a running time of $Tn^2 n_A^2$, which will usually be

significantly smaller than the running time suggested above of TN^2n^2.

4.2.2 Online Estimation for a Group of Disjoint ζ-Independent Sets

Assume now we wish to do this estimate on a set A which is a union of n_s disjoint ζ-independent sets, denoted by A_1, ..., and A_{n_s}. Because each set A_i is ζ-independent, by Corollary 4.1.2 then A is also ζ-independent so we may use the procedure described above. This would take about $n^2n_A^2$ computations per time step. However, there is a better way to do it.

Denote by U_A and U_i the unobserved sites of A and of A_i, respectively. Let B denote the union of all sets A_j in A except A_i: $B = \bigcup_{j\neq i} A_j$. Also let U_B denote the unobserved sites of B. Then we know that

$$P\big(U_A[k] = j \big| O^{0,k}\big) = P\big(U_i[k] = j_i, U_B[k] = j_B \big| O^{0,k}\big) = P\big(U_i[k] = j_i \big| O^{0,k}\big) P\big(U_B[k] = j_B \big| O^{0,k}\big),$$

where j_i and j_B are the states of sets U_i and U_B, respectively, that are consistent with the state j of U_A. Using this reasoning repeatedly, we find that

$$\begin{aligned} P\big(U_A[k] = j \big| O^{0,k}\big) &= \zeta_{A,j}[k] \\ &= \prod_{i=1}^{n_s} P\big(U_i[k] = j_i \big| O^{0,k}\big) \\ &= \prod_{i=1}^{n_s} \zeta_{i,j_i}[k], \end{aligned} \tag{4.2.4.1}$$

where $\zeta_{i,j_i}[k]$ denotes the ζ-variable associated with the set A_i, and $\zeta_{A,j}[k]$ is the ζ-variable of the set A. Therefore, Equation 4.2.4.1 tells us that the ζ-variable of a ζ-independent set that can be decomposed into disjoint ζ-independent sets, can be found by the product of the ζ-variables of the smaller sets. Therefore, if a ζ-independent set can be decomposed into smaller disjoint ζ-independent sets, it is much more efficient to calculate the ζ-variables of the smaller sets, and then simply multiply these to get the ζ-variables of the original set.

Finally, let the state j of the set U_A be also represented by a vector $\mathbf{s}_A = [j_1, ..., j_{n_s}]'$ where j_i is the state of the set A_i consistent with the state j of U_A. Then if the state estimate is $U_A^*[k] = j^*$, we may find the corresponding vector $\mathbf{s}_A = [j_1^*, ..., j_{n_s}^*]'$ entry by entry, by

$$j_i^* = \arg \max_{1 \le j \le n_i} \zeta_{i,j_i}[k], \tag{4.2.4.2}$$

where n_i is the number of states that the set A_i may take: $n_i = \prod_{j \in A_i} m_j$. This is because

$$\begin{aligned} U_A^*[k] &= \arg_j \max_{1 \le j \le n_A} P(U_A[k] = j | O^{0,k}) \\ &= \arg_j \max_{1 \le j \le n_A} \prod_{i=1}^{n_s} \zeta_{i,j_i}[k] \\ &= \arg_{[j_1,...,j_{n_s}]} \prod_{i=1}^{n_s} \left(\max_{1 \le j_i \le n_i} \zeta_{i,j_i}[k] \right). \end{aligned}$$

What this says is that estimating the most likely state of A is the same as estimating the most likely state of each set A_i independently and then combining them.

Therefore, to use this method at each time we need to solve for the ζ-variables of each of the n_s ζ-independent sets. The number of computations per time period will therefore be $n^2 \sum_{i=1}^{n_s} n_i^2$. So using this algorithm for T time steps will take about

$$R = Tn^2 \sum_{i=1}^{n_s} n_i^2 = Tn^2 \sum_{i=1}^{n_s} \left(\prod_{j \in A_i} m_i^2 \right),$$

which will be significantly lower than the running time of the method where we just take A to be a big ζ-independent set. This latter would have a running time of

$$R = Tn^2 n_A^2 = Tn^2 \prod_{j \in A} m_i^2.$$

This suggest a valuable lesson that is often encountered in the area of algorithms, which practice is commonly referred to as *divide and conquer*. Decomposing a big problem into smaller ones and then integrating the solution of each subproblem results in much more efficient algorithms than attacking the big problem directly. In this case, if we wish to

estimate the most likely state of a ζ-independent set in the sense defined in this section, it is best to try to decompose this set first into as many disjoint ζ-independent sets as possible, solve the problem for each one of those sets, and integrate the solutions as described above.

4.2.3 Online Estimation for All the Influencing Sites of the Network

The previous section, when combined with the concept of a ζ-minimal partition of the network graph suggests an efficient method to estimate the most likely state of the set of all influencing sites in the network, denoted by I.

Let the ζ-minimal partition of a network graph $\Gamma[\mathbf{D}']$ be given by $\mathcal{P} = \{\overline{I}, A_1, ..., A_{n_m}\}$. Then we know that the sets A_i for $1 \leq i \leq n_m$ are all disjoint, ζ-minimal, and that they cover the set of influencing sites I of the network. We may therefore use the method of the previous section to estimate the most likely state of the influenced sites at each time given the observations up to that time. Because each A_i set is ζ-minimal, this method results in the largest number of disjoint ζ-independent sets that cover I, which makes the ζ-minimal partition the most efficient way to decompose this problem.

4.2.4 Online Estimation for The Whole Network

In Section 4.2.3 we describe how to efficiently solve the estimation problem that has been addressed in this section, for the set of all influencing sites. Now we wish to estimate the state of the whole network, which means we need to incorporate all the non-influencing sites, denoted by $\overline{I}$, in our estimate. From Section 4.2.2, we know that to make the method as efficient as possible, we should partition the network into ζ-independent, disjoint sets such that the number of these such sets is maximum. We then apply the method described in Section 4.2.2 of finding the ζ-variables for each set at each time, finding the most likely state of each set, and combining the solutions to get the most likely state of the whole network. Therefore, the question that needs to be answered is: How do we find the partition the network graph $\Gamma[\mathbf{D}']$ into ζ-independent, disjoint sets in such a way that the number of these

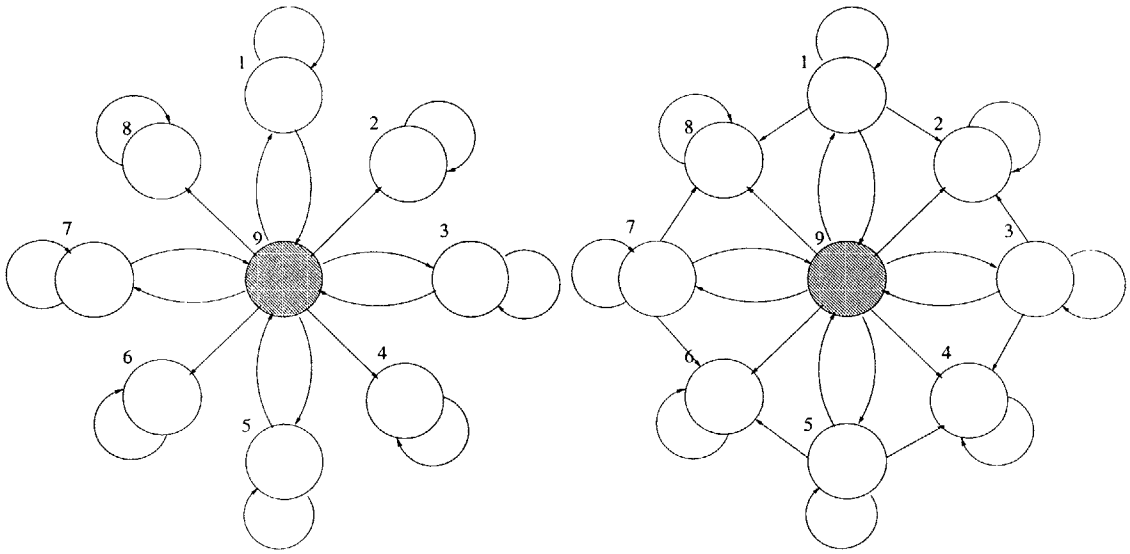

Figure 4-10: Two similar network graphs with very different ζ-network partitions.

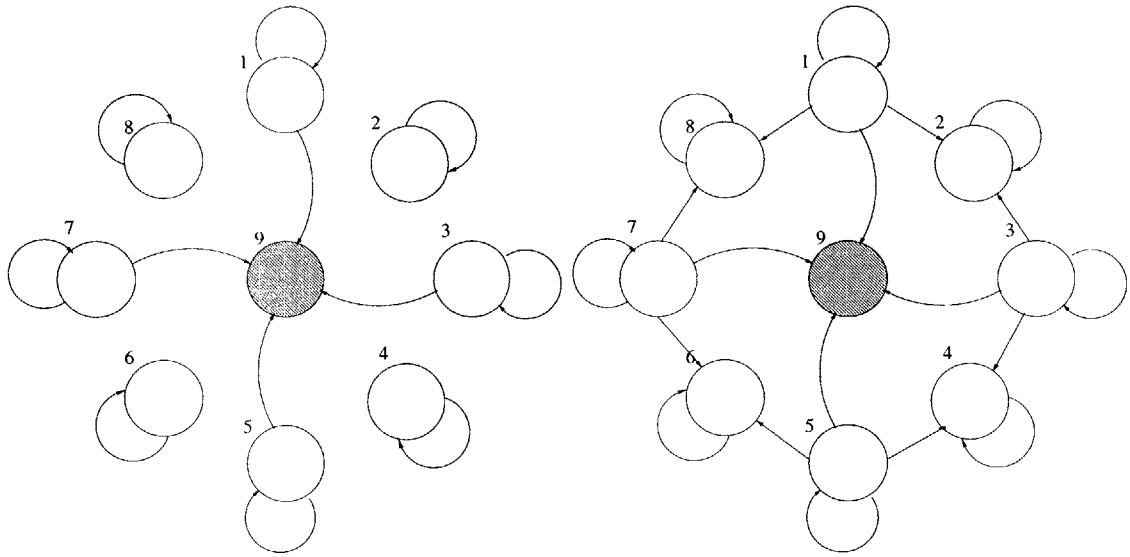

Figure 4-11: The two modified graphs of the networks shown in Figure 4-10. Each disconnected component is a set of the ζ-network partition.

sets is maximum?

It turns out that this partition is the ζ-network partition and may be found by deleting all the edges from the observed sites, and taking each disconnected component of the resulting graph as a ζ-independent set.

The larger the number of sets in the ζ-network partition of a given POIM, the easier computationally it is to estimate the present state of the whole network. Using the same examples shown in Section 4.1.1, we see then that from the two influence graphs on the left and right of figure 4-10, the one on the left is easier to estimate because its ζ-network partition (shown on the left of Figure 4-11) has five sets, as opposed the single one of the other network graph (shown on the right of Figure 4-11).

4.2.5 Online Estimation for an Arbitrary Set of Sites in the Network

Finally, consider we wish to find the estimate of an arbitrary set of unobserved sites U_a, given the observations, just as defined above:

$$U_a^*[k] = \arg \max_{1 \leq i \leq n_a} P\big(U_a[k] = i \big| O^{0,k}\big), \tag{4.2.4.3}$$

where n_a is the number of states that U_a may take. Given our previous discussions, the most efficient way to do this using the concept of ζ-variables is the following:

1. Find the smallest possible ζ-independent set of which U_a is a subset. Call this new set A, and the unobserved sites in it U_A. For this it will be useful to consider sets which are unions of ζ-minimal sets from the ζ-minimal partition of the graph.

2. At each time k, do the following:

   (a) Find the probability distribution $P\big(U_A[k] = j \big| O^{0,k}\big)$ for all possible states j of U_A. This, of course, is done by finding the ζ-variables of set A, $\zeta_{A,j}[k]$. As discussed in Section 4.2.2, if the set A can be broken into disjoint ζ-independent sets then the

ζ-variables should be found by Equation 4.2.4.1. Otherwise we find them directly by Equations 4.1.4.15 and 4.1.4.16.

(b) Find the distribution $P\big(U_a[k] = i \big| O^{0,k}\big)$ from the ζ-variables of A by

$$P\big(U_a[k] = i \big| O^{0,k}\big) = \sum_{\text{j} \mid U_a = i} \zeta_{A,j}[k], \text{ for all i.}$$

(c) Finally, find the estimate for time k by Equation 4.2.4.3.

4.2.6 Example

To illustrate how these methods may be used, let us use them to estimate the state of the POIM of Examples 3.1.2 and 4.2.6, which is shown again in Figure 4-12 below for convenience. We will use the same model parameters as in previous sections, so that

$$\begin{aligned}
\mathbf{D} &= \begin{bmatrix} .5 & .5 & 0 & 0 & 0 \\ .25 & .5 & .25 & 0 & 0 \\ 0 & .25 & .5 & .25 & 0 \\ 0 & 0 & .5 & .5 & 0 \\ 0 & .25 & .25 & .25 & .25 \end{bmatrix}, \\
\mathbf{A} &= \begin{bmatrix} p & 1-p \\ 1-q & q \end{bmatrix}, \text{ and} \\
\pi_i &= \begin{bmatrix} 0 & 1 \end{bmatrix} \text{ for all sites } 1 \leq i \leq 5.
\end{aligned}$$

This is a POIM whose initial state is completely determined, with all sites being in status 2. The only free parameters are thus p and q.

We will use the method discussed in Section 4.2.4 to estimate at each time k, the most likely state of each of the sites of the network given the observations up to that time: $s_i^*[k] = \arg_j \max P(s_i[k] = j | O^{0,k})$. For this purpose we first identify the ζ-network partition of the network which is shown on the right of Figure 4-12. Then, we will work with the

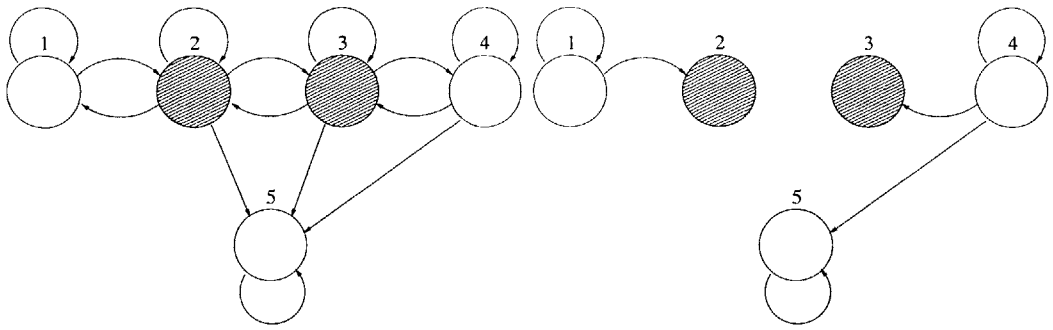

Figure 4-12: Network graph of a homogeneous POIM with 5 sites, and its network partition.

ζ-variables of the sets $A_1 = \{1, 2\}$ and $A_2 = \{3, 4, 5\}$ to find the probability distribution $P(U[k] = j|O^{0,k})$. From this distribution we could estimate $U^*[k]$ by using Equation 4.2.4.2. If this were the case, we would let U_1 and U_2 denote the unobserved sites in A_1 and in A_2, respectively and at each time we would do the estimates of A_1 and A_2 separately by

$$U_i^*[k] = \arg_{j_i} \max \zeta_{i,j_i}[k],$$

and let our estimate of the whole network at time k to be that which is consistent with $U_1^*[k]$ and $U_2^*[k]$. Instead, we wish to estimate each site independently of the others, so we find the distributions $P(s_i[k] = j|O^{0,k})$ for $i = \{1, 4, 5\}$ by summing over the distribution $P(U[k] = j|O^{0,k})$ appropriately.

For a run with 100 time steps and $p = q = .1$, we used the method described above to estimate the state of sites 1, 4, and 5. The results of doing so are shown in Figure 4-13. The number of time steps where the estimate predicts the actual status of the sites is surprisingly high. For this particular run, the estimator predicted the status correctly for sites 1, 4, and 5 a total of 69, 84, and 70 times, respectively, out of 100.

It is interesting to compare the results of these estimates, that use the information included in the observations, to those estimates we would obtain if we did not use the information of the observations. Recall that for a homogeneous POIM, the expected state matrix at

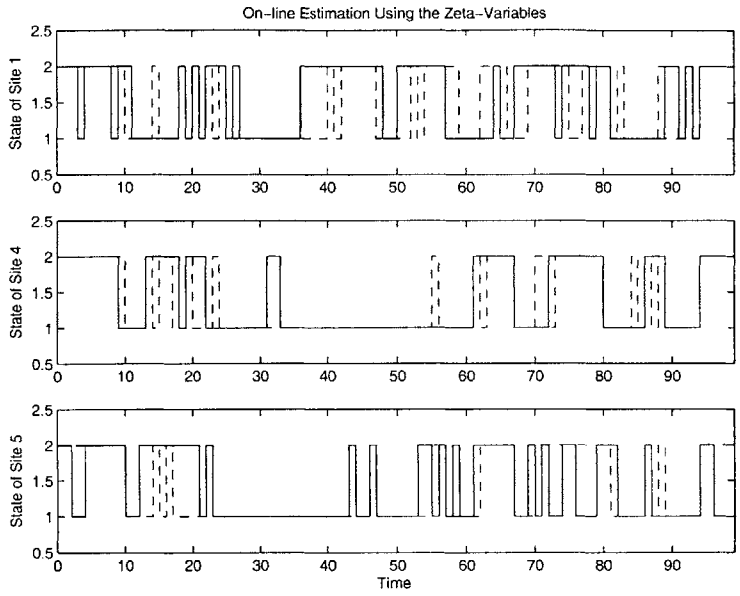

Figure 4-13: Results of doing online Estimation for sites 1, 4 and 5 of the POIM of Figure 4-12. In the three plots, the solid line is the actual state of the given site, and the dashed line is the estimate obtained by the on line estimation method. The upper plot is for site 1, the middle one for site 4, and the lower one for site 5.

time k given the initial state matrix $\mathbf{S}[0]$ is given by $E[\mathbf{S}[k]] = \mathbf{D}^k[\mathbf{S}[0]\mathbf{A}^k$. The ith row of $E[\mathbf{S}[k]]$ is then the probability distribution of site i at time k. The status that corresponds to the maximum valued entry of row i is therefore the most likely status of site i at time k. Doing these for sites 1, 4 and 5 for all times we come up with another estimate of their state. We did this for the data shown in Figure 4-12 and found that the number of hits for this estimator was 56, 41, and 41 for sites 1, 4, and 5, respectively, out of the 100 time steps. Clearly, there is value in incorporating the observations into our estimate.

Exploring this a little further, we compared the performance of these two estimates (one using the observations, one without using them) for different values of p and q on the POIM above. For each set of values, we created 1000 runs of 100 time steps each. For each run, we computed the number of hits of both estimators, and computed the mean number of hits for

Parameters (p,q)	(.1,.1)	(.2,.2)	(.3,.3)	(.8,.8)	(.9,.9)	(.45,.55)
On Line Hits	77.9750	67.9780	61.0870	68.0830	77.7880	55.5780
Expected Hits	51.7510	51.0640	51.0160	51.3460	52.4890	55.5780
On Line Std Dev	5.1959	5.1674	5.2010	5.2061	5.1819	4.8007
Expected Std Dev	10.7725	7.7938	5.3442	7.6170	11.1897	4.8007

Figure 4-14: This table compares the estimates of the status of site 1 using and without using the observations.

Parameters (p,q)	(.1,.1)	(.2,.2)	(.3,.3)	(.8,.8)	(.9,.9)	(.45,.55)
On Line Hits	77.8660	67.9950	61.4330	68.0780	78.0490	55.4250
Expected Hits	52.5730	51.2420	50.8300	50.8800	52.2720	55.4250
On Line Std Dev	5.1866	4.9487	5.1328	5.1651	5.3441	4.7113
Expected Std Dev	11.4006	7.7999	5.0619	7.4572	10.8412	4.7113

Figure 4-15: This table compares the estimates of the status of site 4 using and without using the observations.

Parameters (p,q)	(.1,.1)	(.2,.2)	(.3,.3)	(.8,.8)	(.9,.9)	(.45,.55)
On Line Hits	72.1570	62.0990	56.8400	62.3780	72.3580	55.2930
Expected Hits	52.0790	50.7480	50.8580	51.3680	52.3920	55.2930
On Line Std Dev	5.4906	5.2365	5.2335	5.4006	5.3267	4.8015
Expected Std Dev	9.4540	6.5329	5.0697	6.1697	9.2566	4.8015

Figure 4-16: This table compares the estimates of the status of site 5 using and without using the observations.

sites 1, 4 and 5 of each estimator and the standard deviation. The results are summarized in Tables 4-14, 4-15, and 4-16. Notice that for values of p and q away from .5 the value of the observation is considerable, and the on line estimator does much better than the just using the expected state without the observations. But as p and q approach .5, the number of hits of the two estimators becomes the same. In fact, the two estimators converge as p and q reach .5. This says that the value of the information given by the observations is a function of the parameters p and q.

4.3 Finding the Probability of an Observation Sequence

In this section we go back to the problem of finding the probability of an observation sequence in a POIM with a general influence graph $\Gamma[\mathbf{D}']$. The solution we present here works with ζ-independent sets and is much more efficient than that presented in Section 3.1.1.

We first note that the probability of an observation sequence from time 0 to time $k+1$, can be decomposed into a product of $k+1$ terms:

$$\begin{aligned} P(O^{0,k+1}) &= P(O^0)P(O^{1,k+1}|O^0) \\ &= P(O^0)P(O^1|O^0)P(O^{2,k+1}|O^{0,1}) \\ &= P(O^0)P(O^1|O^0)P(O^2|O^{0,1})P(O^{3,k+1}|O^{0,2}) \\ &= P(O^0)\prod_{l=0}^{k} P(O^{l+1}|O^{0,l}). \end{aligned} \tag{4.3.4.1}$$

From the definition of a POIM we know that $P(O^0) = \prod_{i\in O}\pi_i[j_i]$, where j_i is the status of site i consistent with the observations at time 0. To find $P(O^{0,k+1})$ via Equation 4.3.4.1 we need to find the terms $P(O^{l+1}|O^{0,l})$ for $0 \leq l \leq k$. We do this with the help of the ζ-minimal partition.

Consider the ζ-minimal sets of the ζ-minimal partition of the network graph $\Gamma[\mathbf{D}']$ that contain some observed sites in them. We label these sites A_1,..., and A_{n_o}, and we refer to the ζ-variable of set A_i as $\zeta_{i,j}[k]$, where the unobserved sites of A_i are in the state j. Denote the unobserved and observed sites of site A_i by U_i and O_i, respectively. Since each set A_i is ζ-independent, then we get the following relation using Equation 4.1.4.12 for each ζ-independent set A_i:

$$\begin{aligned} P\big(O^{l+1}\big|O^{0,l}\big) &= P\big(O_1^{l+1},...,O_{n_o}^{l+1}\big|O^{0,l}\big) \\ &= P\big(O_1^{l+1}\big|O^{0,l}\big)P\big(O_2^{l+1},...,O_{n_o}^{l+1}\big|O^{0,l}\big) \\ &= \prod_{i=1}^{n_o} P\big(O_i^{l+1}\big|O^{0,l}\big). \end{aligned} \tag{4.3.4.2}$$

Finally, we may find the probability $P\big(O_i^{l+1}\big|O^{0,l}\big)$ for each set A_i from its ζ-variables at time l by

$$P\big(O_i^{l+1}\big|O^{0,l}\big) = \sum_{j=1} P(O_i^{l+1}|O^l, U_{A_i}[l]=j)\zeta_{i,j}[l].$$

In fact, $P\big(O_i^{l+1}\big|O^{0,l}\big)$ is the denominator of Equation 4.1.4.16, so we calculate these probabilities anyways when finding the ζ-variables for the set A_i. Therefore, finding the probability of the observation sequence from time 0 to time T, requires only finding the ζ-variables for the sets A_i,..., and A_{n_o} obtained from the ζ-minimal partition from times 0 to T. At each time k and for each set A_i we simply store the probabilities $P\big(O_i^k\big|O^{0,k-1}\big)$ and we find the probability of the observation sequence by

$$P\big(O^{0,T}\big) = P(O^0)\prod_{k=1}^{T}\Big(\prod_{i=1}^{n_o} P\big(O_i^k|O^{0,k-1}\big)\Big).$$

We may also find the probability of an observation sequence from 0 until l, with $l \leq T$ by

$$P\big(O^{0,l}\big) = P(O^0)\prod_{k=1}^{l}\Big(\prod_{i=1}^{n_o} P\big(O_i^k|O^{0,k-1}\big)\Big).$$

Therefore the order of the computations required by this method is dominated by the procedure of finding the ζ-variables for the sets A_i. Assume each set A_i can take n_i possible states, with $n_i = \prod_{j\in U_i} m_i$. Then calculating the ζ-variables up to time T for the set A_i will take on the order of $Tn^2n_i^2$ computations. Therefore this method will take on the order of $Tn^2\sum_{i=1}^{n_o} n_i^2$ computations, which is significantly better than the number of computations of the method in Section 3.1.1.

This method of finding $P(O^{0,k})$ by using the ζ-variables of the ζ-minimal sets that cover the set of observed sites, works in exactly the same way if we choose any other number of ζ-independent sets that cover the observed sites. But the sets in the ζ-minimal partition are the smallest possible ones so the computation of their ζ-variables is easier. However, if we are interested in estimating the state of the network and in finding the likelihood of an observation sequence $P(O^{0,k})$, we may use the same ζ-independent for both purposes.

4.3.1 Example

Consider the usual homogeneous POIM with network graph as shown on the left Figure of 4-17, and the same model paramters as those used in Section 4.2.6. To find the probability of an observation sequence, we need to find the ζ-minimal sets from the ζ-minimal partition that contain all the observed sites. In this case, the relevant sets are those formed by sites 1 and 2, and by 3 and 4, as shown on the figure on the right. To calculate the probability of an observation sequence, we find the ζ-variables of these two sets and use Equation 4.3.4.3.

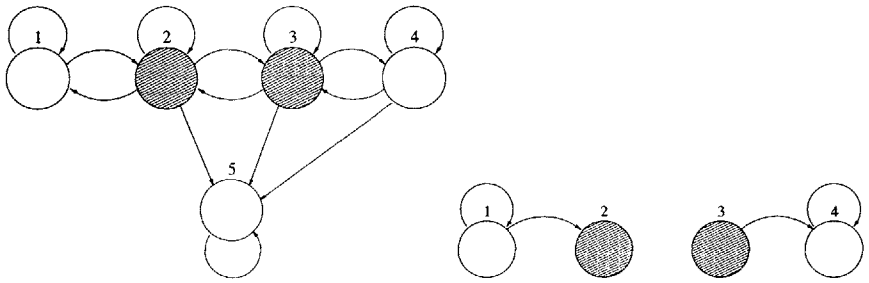

Figure 4-17: The figure on the left shows the original network graph $\Gamma[\mathbf{D}']$. The figure on the right shows the two ζ-minimal sets that result from the ζ-minimal partition of the graph and that cover the set of observed sites.

For $p = .1$ and $q = .1$, we computed the probability of the possible observation sequences for 3 time steps. Since the initial state is fixed, then we may have have 16 possible observation

Observation	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8
Likelihood	0.0117	0.0150	0.0855	0.0138	0.0150	0.0052	0.0318	0.0020
Observation	O_9	O_{10}	O_{11}	O_{12}	O_{13}	O_{14}	O_{15}	O_{16}
Likelihood	0.0855	0.0318	0.6273	0.0294	0.0138	0.0020	0.0294	0.0008

Figure 4-18: This table shows the likelihood of the possible observation sequences over a 3 step run of the POIM in Figure 4-17.

sequences, namely:

$$
\begin{aligned}
O_1 &= \{s_2 = 211, s_3 = 211\} \\
O_2 &= \{s_2 = 221, s_3 = 211\} \\
O_3 &= \{s_2 = 212, s_3 = 211\} \\
O_4 &= \{s_2 = 222, s_3 = 211\} \\
O_5 &= \{s_2 = 211, s_3 = 221\} \\
O_6 &= \{s_2 = 212, s_3 = 221\} \\
O_7 &= \{s_2 = 221, s_3 = 221\} \\
O_8 &= \{s_2 = 222, s_3 = 221\} \\
O_9 &= \{s_2 = 211, s_3 = 212\} \\
O_{10} &= \{s_2 = 221, s_3 = 212\} \\
O_{11} &= \{s_2 = 212, s_3 = 212\} \\
O_{12} &= \{s_2 = 222, s_3 = 212\} \\
O_{13} &= \{s_2 = 211, s_3 = 222\} \\
O_{14} &= \{s_2 = 221, s_3 = 222\} \\
O_{15} &= \{s_2 = 212, s_3 = 222\} \\
O_{16} &= \{s_2 = 222, s_3 = 222\}.
\end{aligned}
$$

Using these method, we found the likelihood of each one of these observation sequences. The results are summarized in Table 4-18.

4.4 The Forward and Backward Variables: Local Decoding

In this section we describe how to take advantage of the graph structure of the network to find the forward and backward variables introduced in Section 3.1.1, in a much more efficient way. These variables are needed to do local decoding, which is described in Section 3.2.1, and consists of estimating the state of the network at time k, with $0 \leq k \leq T$, given the observations from 0 to T.

Recall that the forward and backward variables (α and β, respectively) are defined by

$$\begin{aligned} \alpha_i[k] &= P\big(U[k] = i, O^{0,k}\big), \qquad \text{and} \\ \beta_i[k] &= P\big(O^{k+1,T} \big| U[k] = i,\ O^k\big), \end{aligned}$$

where U and O are the unobserved and observed sites of the network.

4.4.1 The Forward Variables

In Section 4.2.4 we discussed how to find the ζ-variables of the whole network by using the ζ-network partition. On the other hand, we discussed in Section 4.3 how to find the probability of the observations from 0 to k. We may thus find the forward variables from the following relation:

$$\begin{aligned} \alpha_i[k] &= P\big(U[k] = i, O^{0,k}\big) \\ &= P\big(U[k] = i \big| O^{0,k}\big) P\big(O^{0,k}\big) \\ &= \zeta_i[k] P\big(O^{0,k}\big). \end{aligned}$$

4.4.2 The Backward Variables

The story for the backward variables is a little more elaborate. There are two main observations that allow us to take advantage of the graph structure. The first is that since the non-influencing sites will never influence the observed sites, then the backward variables are

independent of the state of the non-influencing sites at all times:

$$\begin{aligned}\beta_i[k] &= P\big(O^{k+1,T}\big|U[k]=j_i,\ O^k\big)\\ &= P\big(O^{k+1,T}\big|I[k]=i,\ O^k\big),\end{aligned}$$

where I denotes the influencing sites of the network, and the state i of set I is consistent with the state j_i of the set U_i. This observation means we can simply forget about the non-influencing sites when calculating the backward variables. The second observations is that if the network graph $\Gamma[\mathbf{D}']$ consists of two or more disconnected components, the backward variable of each may be found separately of the rest, and the backward variable of the whole network is just the product of the backward variables of the disconnected components.

Combining these two observations we propose the following method to find the backward variables:

1. Split the influencing sites into disjoint sets as follows:

   (a) Erase all the non-influencing sites from the network graph $\Gamma[\mathbf{D}']$ along with any arrows to and from them.

   (b) Assume there are n_b disconnected components in the resulting graph. Label these sets $B_1,...,B_{n_o}$. Let O_i and U_i denote the observed and unobserved sites in B_i. Also let $n_i = \prod_{j\in U_i} m_j$ be the number of states that the set U_i may take.

2. For each set B_i define the backward variable $\beta_{i,j}[k]$ associated with it, by

$$\beta_{i,j}[k] = P\big(O_i^{k+1,T}\big|U_i[k]=j, O_i^k\big).$$

3. Find the backward variables for each set B_i independently of the others by the following algorithm:

   (a) Initialization:

$$\beta_{i,j}[T] = 1 \qquad \forall j$$

(b) Recursion:

$$\beta_{i,j}[k] = \sum_{l=1}^{n_i} P\big(U_i[k+1] = l, O_i^{k+1} \big| U_i[k] = j, O_i^k\big)\beta_{i,l}[k+1] \qquad \forall j, \text{ for } 0 \leq k < T.$$

4. Find the backward variables of the whole network by:

$$\beta_j[k] = \prod_{i=1}^{n_o} \beta_{i,j_i}[k], \qquad \forall j, 0 \leq k \leq T,$$

where j_i is the state of the set U_i consistent with the state j of the influencing sites of the network.

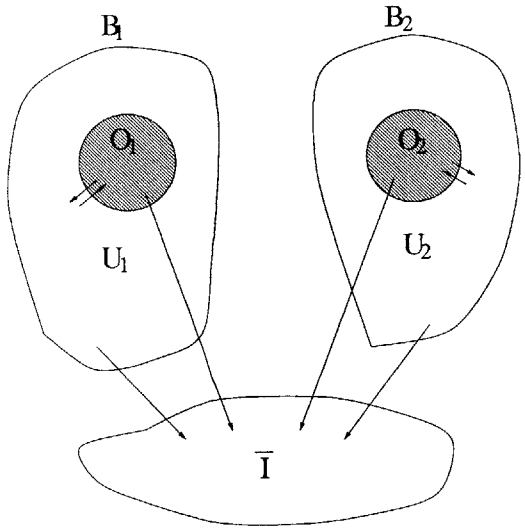

Figure 4-19: This figure illustrates a network graph that consists of the non-influencing sites and two more sets of sites, B_1 and B_2, that have no incoming arrows and whose only outgoing arrows are to non-influencing sites.

Now we show why we may split up the calculation of the backward variables of the network into the sets described above. We show this for the case were the influencing sites in the network can be decomposed into two sets, B_1 and B_2. The case with more sets is a direct

extension of the argument below. Consider the network graph in Figure 4-19, where the sets B_1 and B_2 are disjoint once we remove the non-influencing sites from the network graph. Let O and U denote the observed and unobserved sites in the network, and let O_1 (O_2) and U_1 (U_2) be the observed and unobserved sites in B_1 (B_2). We wish to show that

$$P\big(O^{k+1,T}\big|U[k]=j,O^k\big)=P\big(O_1^{k+1,T}\big|U_1[k]=j_1,O_1^k\big)P\big(O_2^{k+1,T}\big|U_2[k]=j_2,O_2^k\big),$$

where j_1 (j_2) is the state of U_1 (U_2) that is consistent with state j of the unobserved sites of the network.

We do this by induction. For $k=T$, we have that

$$\begin{aligned}\beta_j[T] &= P\big(O^{T+1,T}\big|U[T]=j,O^T\big)\\ &= P\big(\emptyset\big|U[T]=j,O^T\big)\\ &= 1\\ &= P\big(O_1^{T+1,T}\big|U_1[T]=j_1,O_1^T\big)P\big(O_2^{T+1,T}\big|U_2[T]=j_2,O_2^T\big).\end{aligned}$$

Now assume that

$$P\big(O^{k+2,T}\big|U[k+1]=l,O^{k+1}\big)=P\big(O_1^{k+2,T}\big|U_1[k+1]=l_1,O_1^{k+1}\big)P\big(O_2^{k+2,T}\big|U_2[k+1]=l_2,O_2^{k+1}\big),$$

so that

$$
\begin{aligned}
\beta_j[k] &= P\big(O^{k+1,T}\big|U[k]=j,O^k\big)\\
&= \sum_l P\big(O^{k+2,T}\big|I[k+1]=l,O^{k+1}\big)P\big(I[k+1]=l,O^{k+1}|U[k]=j,O^k\big)\\
&= \sum_l P\big(O_1^{k+2,T}\big|U_1[k+1]=l_1,O_1^{k+1}\big)P\big(O_2^{k+2,T}\big|U_2[k+1]=l_2,O_2^{k+1}\big)\\
&\quad\times P\big(I[k+1]=l,O^{k+1}|U[k]=j,O^k\big)\\
&= \sum_{l_1}\sum_{l_2} P\big(O_1^{k+2,T}\big|U_1[k+1]=l_1,O_1^{k+1}\big)P\big(O_2^{k+2,T}\big|U_2[k+1]=l_2,O_2^{k+1}\big)\\
&\quad\times P\big(U_1[k+1]=l_1,O_1^{k+1}|U_1[k]=j_1,O_1^k\big)P\big(U_2[k+1]=l_2,O_2^{k+1}|U_2[k]=j_2,O_2^k\big)\\
&= \sum_{l_1}\Big(P\big(O_1^{k+2,T}\big|U_1[k+1]=l_1,O_1^{k+1}\big)P\big(U_1[k+1]=l_1,O_1^{k+1}|U_1[k]=j_1,O_1^k\big)\Big)\\
&\quad\times\sum_{l_2}\Big(P\big(O_2^{k+2,T}\big|U_2[k+1]=l_2,O_2^{k+1}\big)P\big(U_2[k+1]=l_2,O_2^{k+1}|U_2[k]=j_2,O_2^k\big)\Big)\\
&= \sum_{l_1}\Big(\beta_{1,l_1}[k+1]P\big(U_1[k+1]=l_1,O_1^{k+1}|U_1[k]=j_1,O_1^k\big)\Big)\\
&\quad\times\sum_{l_2}\Big(\beta_{2,l_2}[k+1]P\big(U_2[k+1]=l_2,O_2^{k+1}|U_2[k]=j_2,O_2^k\big)\Big)\\
&= P\big(O_1^{k+1,T}\big|U_1[k]=j_1,O_1^k\big)P\big(O_2^{k+1,T}\big|U_2[k]=j_2,O_2^k\big).
\end{aligned}
$$

The fact that we may split up the calculations of the forward and backward variables of the network into sets of sites might suggest that localized decoding on the network might be done by finding the most likely state of different sets of the network independently and then putting them together. This would be the case if the sets used to compute the forward and backward variables were the same. But they are not. So to find the most likely state of the network at at time $k \neq T$, given the observations from 0 to T we must consider all the possible states that the unobserved sites as a whole may take. This is unfortunate. On the other hand, we are able to do this for the estimation problem of Section 4.2.

4.5 The Viterbi Variables: Global Decoding

In this section we describe how to take advantage of the graph structure of the network to find the Viterbi variables, introduced in Section 3.2.2, in a much more efficient way. These variables are needed to do global decoding which consists of estimating the state sequence from time 0 to T, given an observation sequence in that time range.

Recall that the Viterbi variables δ are defined by

$$\delta_i[k] = \max_{U^{0,k-1}} P\big(O^{0,k}, U^{0,k-1}, U[k] = i\big),$$

where U and O are the unobserved and observed sites of the network, respectively.

It turns out that the Viterbi variables for a set may be found independently of the rest of the network if the set is such that there are no incoming arrows into it, and the only outgoing arrows are to non-influencing sites.

Consider such a set A and its associated basic partition. We define the Viterbi variables for set A as

$$\delta_i^A[k] = \max_{U_A^{0,k-1}} P\big(O_A^{0,k}, U_A^{0,k-1}, U_A[k] = i\big).$$

These can be used to do global decoding for set A. Letting

$$q_{ji}^A[k] = P(O_A^{k+1}, U_A[k+1] = i | O_A^k, U_A[k] = j)$$

for convenience, the Viterbi variables for set A can be found by the following algorithm:

1. Initialization:

$$\begin{aligned} \delta_i^A[0] &= \prod_{j \in A} \pi_j[i_j] \qquad \text{for } 1 \le i \le N_A \\ \psi_i^A[0] &= 0 \qquad \text{for } 1 \le i \le N_A. \end{aligned}$$

Here, i_j denotes the status of site j consistent with the state i of the unobserved sites in A. The number of states that the unobserved sites in A can take is $N_A = \prod_{i \in U_A} m_i$.

2. Iteration:

$$\begin{aligned} \delta_i^A[k+1] &= \max_{1 \le j \le N_A} \delta_j^A[k] q_{ji}^A[k] \qquad \text{for } 1 \le i \le N_A \text{ and } 1 \le k \le T \\ \psi_i[k+1] &= \arg \max_{1 \le j \le N_A} \delta_j[k] q_{ji}^A[k] \qquad \text{for } 1 \le i \le N_A \text{ and } 1 \le k \le T. \end{aligned}$$

3. Termination:

$$\begin{aligned} U_A^{*T} &= \arg \max_{1 \le i \le N_A} \delta_i^A[T] \\ U_A^{*k} &= \psi(U_A^{*k+1}) \qquad \text{for } T-1 \ge t \ge 0. \end{aligned}$$

Therefore it is relatively easy to do global decoding for a set A that has no incoming arrows and outgoing arrows to non-influencing sites only, since this algorithm takes on the order of TN_A^2 computations. It is also that the Viterbi variables of a set that can be partitioned into subsets that satisfy the conditions described above are the product of viterbi variables of the subsets. The following proof of the algorithm stated above will demonstrate this.

Consider a set A with no incoming arrows and outgoing arrows to non-influencing sites only. Consider its associated basic partition (Definition 2). Let C be the union of A and B. We want to show that $\delta_i^C[k] = \delta_{i_A}^A[k] \delta_{i_B}^B[k]$ for all times k, where i_A (i_B) is the state of the unobserved sites in A (B) that is consistent with the state i of the unobserved sites in C. Doing this we will also prove the algorithm above.

We prove this by induction. At time 0, we have that

$$\begin{aligned} \delta_i^C[0] &= P\big(O^0, U_C[0] = i\big) \\ &= P\big(O_A^0, U_A[0] = i\big) P\big(O_B^0, U_B[0] = i\big) \\ &= \delta_{i_A}^A[0] \delta_{i_B}^B[0]. \end{aligned}$$

Now assuming that $\delta_i^C[k] = \delta_{i_A}^A[k]\delta_{i_B}^B[k]$, we have that

$$\begin{aligned}
\delta_i^C[k+1] &= \max_{U_C^{0,k}} P\big(O_C^{0,k+1}, U_C^{0,k}, U_C[k+1] = i\big) \\
&= \max_{U_C^{0,k-1}} \max_j P\big(O_C^{0,k+1}, U_C^{0,k-1}, U_C[k] = j, U_C[k+1] = i\big) \\
&= \max_{U_C^{0,k-1}} \max_j P\big(O_C^{0,k}, U_C^{0,k-1}, U_C[k] = j\big) P\big(O_C^{k+1}, U_C[k+1] = i | O_C^k, U_C[k] = j\big) \\
&= \max_j \big[\max_{U_C^{0,k-1}} P\big(O_C^{0,k}, U_C^{0,k-1}, U_C[k] = j\big)\big] P\big(O_C^{k+1}, U_C[k+1] = i | O_C^k, U_C[k] = j\big) \\
&= \max_j \delta_j^C[k]\ P\big(O_C^{k+1}, U_C[k+1] = i | O_C^k, U_C[k] = j\big) \\
&= \max_{j_A, j_B} \delta_{j_A}^A[k]\ \delta_{j_B}^B[k]\ P\big(O_C^{k+1}, U_C[k+1] = i | O_C^k, U_C[k] = j\big) \\
&= \max_{j_A, j_B} \big[\delta_{j_A}^A[k]\ \delta_{j_B}^B[k]\ P\big(O_A^{k+1}, U_A[k+1] = i_A | O_A^k, U_A[k] = j_A\big) \\
&\quad \times P\big(O_B^{k+1}, U_B[k+1] = i_B | O_B^k, U_B[k] = j_B\big)\big] \\
&= \big(\max_{j_A} \delta_{j_A}^A[k] q_{j_A i_A}^A[k]\big)\big(\max_{j_B} \delta_{j_B}^B[k]\ q_{j_B i_B}^B[k]\big) \\
&= \delta_{i_A}^A[k+1]\delta_{i_B}^B[k+1].
\end{aligned}$$

The last inequality follows from the fact that

$$\begin{aligned}
\max_{j_A} \delta_{j_A}^A[k] q_{j_A i_A}^A[k] &= \max_{j_A} \max_{U_A^{0,k-1}} \big[P\big(A^{0,k-1}, O_A^k, U_A[k] = j_A\big) \\
&\quad \times P\big(O_A^{k+1}, U_A[k+1] = i_A | O_A^k, U_A[k] = j_A\big)\big] \\
&= \max_{j_A} \max_{U_A^{0,k-1}} P\big(A^{0,k-1}, O_A^{k,k+1}, U_A[k], U_A[k+1] = i_A\big) \\
&= \max_{U_A^{0,k}} P\big(A^{0,k}, O_A^{k+1}, U_A[k+1] = i_A\big) \\
&= \delta_{i_A}^A[k+1],
\end{aligned}$$

along with a symmetric statement that can be made for set B.

Chapter 5

Conclusion

In this thesis we introduced the partially observable influence model (POIM), and have developed techniques to make it a useful stochastic model. The influence model is a mathematical representation of stochastic interactions on networks. It comprises a graph where each node (or site) is like a Markov chain whose transition probabilities depend on its present state and that of its neighbors. The POIM is an underlying influence model where the state of the network is partially known. At each time, only the state of a some nodes is known. We call these nodes observed and refer to their state as the observation.

The results developed on this thesis study methods to find the probability of a given observation sequence, estimate the state of the network (and of sets of nodes within the network) given an observation sequence, and then attempt to find the parameters that define a homogeneous POIM based on a sequence of observations.

After we introduce the thesis in Chapter 1, we present the background material and introduce the POIM in Chapter 2. This background material is a review on hidden Markov models, and on the influence model. We also formally formulate the problems that the methods developed in this thesis aim to address.

In Chapter 3 we propose methods, analogous to those used for hidden Markov models, to address the problems of interest. We develop algorithms that find the probability of a given

observation sequence, and the most likely state of the network (and of sets of nodes within the network) given an observation sequence exactly. These algorithms are the Forward-backward and the Viterbi applied to a POIM. However the running time of this algorithms grows as the square of the number of possible states that the network may take. This number grows exponentially as we add nodes to the network, so the methods presented will be useful only for networks with a small number of nodes (or with a small state space). We also study a method to find the parameters of a POIM based solely on the observations. The method we propose works only for POIMs that have a homogeneous influence model at its core. It will not return the correct answer every time, and it has a horrendous running time. However, there is an conceptually simple way to check if the answer returned by the algorithm is correct. Even if it does not return the right answer, it may be used iteratively to find parameters that continually increase the likelihood of the observation.

The methods presented in Chapter 3 do not take advantage of the graph structure of the network. In Chapter 4 we analyze this structure and find much more efficient methods to find the probability of a given observation sequence, and the most likely state of the network (and of sets of nodes within the network) given an observation sequence. The general idea is to break each problem into as small a number of problems as the graph structure allows, solve each subproblem independently of the others, and finally integrate the answer of all the subproblems to obtain the solution of the original problem. For this purpose we have to introduce several concepts. We start by classifying the unobserved sites into influencing and non-influencing. The influencing sites are those whose state at any given time has the potential to influence the state of one or more observed sites in the future. The non-influencing sites are the ones that may not influence any observed site. We also introduce the concepts of the ζ-variables (read zeta variables), and study what network properties allow for the the ζ-variables for some sets to be calculated recursively, and independently from the rest of the network. We call these sets ζ-independent. We then study ways to partition the network into ζ-independent sites. This is where we introduce the ζ-minimal and the ζ-network partitions of the network. These partitions consist of disjoint sets that cover the whole network graph.

We then study how to do estimation of the present state of a set of nodes, or of the whole network, given the observations until the present. We do this by finding the smallest ζ-independent set that includes the original set we were interested in, splitting it into as many ζ-independent sets as possible, doing the estimation for each smaller set separately and finally integrating the estimates from the smaller sets to find the estimate of the original set. We use the ζ-minimal partition to produce estimates for the set of all influencing sites in the network and the ζ-network partition to produce estimates for the whole network. The running time of these algorithms grows as a sum of squares of the state space sizes for the ζ-independent sets involved. This makes their running time at least as good as that of the methods presented in Chapter 3, and potentially significantly better.

We then proceed to use the ζ-minimal partition to find the probability of an observation sequence. The running time of this method is again as good as that for the method presented in Chapter 3, and will most likely be significantly better. Roughly, the difference in performance grows exponentially with the number of non-influencing sites in the graph, and with the number of minimal sets in the ζ-minimal partition.

Finally we show how to find the forward, backward, and Viterbi variables by taking advantage of the graph structure.

We hope that these methods will make the use of the influence model as a modeling tool for interactions on networks more amenable.

5.1 Future Work

Some directions of future work might be the following:

(1) Study the class of stochastic dynamical systems for which the algorithms developed can be used. For example, the ζ-variables can be defined and found just as easily on any networked stochastic model that evolves in discrete time steps, and where the update of each site in the network is independent of the other sites conditioned on the

present state of the system. In particular, the algorithms presents here work for any Partially Observed Markov Network (POMaN), which is a dyanical system with the following characteristics:

- It evolves in discrete time steps.
- At any given time k, it has a state that we denote by $S[k]$ which is represented by a collection of state variables $s_1[k], ..., s_n[k]$. State variable s_i can take states $1, ..., m_i$, so that s_i takes on one of m_i states at each time and the system as a whole can take $N = \prod_{i=1}^{n} m_i$ states.
- The states of any two state variables at time 1 are independent. We denote the probablity that s_i at time 1 is in its jth state by π_{ij}.
- We have measurements of a fixed set of state variables at each time step. We refer to the measurements as *observations* and to the set of observed state variables as O. We write a sequence of observations from time 1 to k as $O^{1,k}$ and the observations at time k as O^k.
- It is Markov, so that the distribution for the state at time $k+1$ conditioned on the immediately previous state, is independent of all past states: $P\big(S[k+1]|S[1], ..., S[k]\big) = P\big(S[k+1]|S[k]\big)$.
- The transition probabilities $P\big(S[k+1]|S[k]\big)$ factor into n terms, one for each state variable so that $P\big(S[k+1]|S[k]\big) = \prod_{i=1}^{n} P\big(s_i[k+1]|S[k]\big)$. The form of the probabilities $P\big(s_i[k+1]|S[k]\big)$ is assumed to be known, but is otherwise arbitrary.

In the most general case, the probability distribution of s_i at the next time step is a function of all the state variables at the present time. However, in most cases it will be a function of only some variables at the present time. Each of these variables is a *parent* of s_i, and we group them in a set of parents P_i, so that $P\big(s_i[k+1]|S[k]\big) = P\big(s_i[k+1]|P_i[k]\big)$, where $P_i[k]$ denotes the state of the parents of s_i at time k. Similarly, if s_j is a parent of s_i then s_i is a child of s_j. We write as C_i the set of children of s_i. In general, if A is a set of state variables, we denote its state at time k by $A[k]$ and the union the of all parents of the variables in A by P_A.

We are currently writing a paper generalizing these algortihms for POMaNs and exploring the relations between these and Dynamic Bayes Nets (DBNs).

(2) Study better methods for learning the parameters of a POIM. For the particular situation where all the nodes in the network are observed numerical optimization methods can be explored to find the parameters that maximize the likelihood function. This function is highly non-linear and the parameters must satisfy a series of linear constraints, since they are entries of row-stochastic matrices or pmfs. For the general POIM, one approach to learn the parameters may use the EM algorithm, specialized to a POIM, to come up with a local maximum of the likelihood function. However, unless ways are found to take advantage of the graph structure, the running time of each iteration of the algorithm would be at best polinomial in the number of states that the network can take.

(3) Explore the use of the POIM or of POMaNs and the algorithms presented in this thesis as modelling tools of relevant systems.

Bibliography

[1] Chalee Asavathiratham. *The Influence Model: A Tractable Representation for the Dynamics of Networked Markov Chains.* PhD dissertation, MIT, Department of Electrical Engineering and Computer Science, October 2000.

[2] Chalee Asavathiratham, Sandip Roy, Bernard Lesieutre, and George Verghese. The influence model. *IEEE Control Systems*, 21(6):52+, December 2001.

[3] Pierre Baldi and Soren Brunak. *Bioinformatics: The Machine Learning Approach.* The MIT Press, second edition, 2001.

[4] Sumit Basu, Tanzeem Choudhury, Brian Clarkson, and Alex Pentland. Learning human interactions with the influence model. 2001.

[5] Eddy S. Krogh A. Durbin, R. and G. Mitchison. *Biological Sequence Analysis.* Cambridge University Press, first edition, 1998.

[6] Yariv Ephraim and Neri Merhav. Hidden markov processes. *IEEE Transactions on Information Theory*, 48(6):1518+, June 2002.

[7] Warren J. Ewens and Gregory R. Grant. *Statistical Methods in Bioinformatics.* Springer-Verlag, first edition, 2002.

[8] Iain L. McDonald and Walter Zucchini. *Hidden Markov and Other Models for Discrete-valued Time Series.* Chapman & Hall, first edition, 1997.

[9] Lawrence Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceeding of the IEEE*, 77(2):257+, February 1989.

[10] YongHong Tian, TieJun Huang, and Wen Gao. Quantitatively evaluating the influence of social interactions in the community-assisted digital library. *Proceeding of the second ACM/IEEE-CS joint conference on Digital Libraries*, 2002.