The Age of Information in Situation Awareness Networks

Master internship report MN-420699

Jori Selen* 0637922
Supervisors Ivo Adan*, Lachlan Andrew†, Yoni Nazarathy†‡, Hai Le Vu †

September 22, 2012

*Manufacturing Networks Group, Department of Mechanical Engineering, Eindhoven University of Technology, Eindhoven, Netherlands

†Centre for Advanced Internet Architectures, Faculty of Information and Communication Technologies, Swinburne University of Technology, Melbourne, Australia

‡School of Probability and Statistics, The University of Queensland, Brisbane, Australia

Abstract

In this paper we derive analytical distributions up to a bounding box of the age of information in situation awareness networks. Two special cases of these networks are studied which leads to the derivation of two algorithms. We find that each node in a tree structured network has a generalized negative binomial marginal distribution. Finally, we consider two branches of nodes leading from source to a sink and design an algorithm (using the previous algorithms) that is capable of obtaining joint and marginal distributions of these nodes. The algorithm iteratively computes segments of the total joint distribution and therefore keeps time and memory usage manageable.

Contents

Chapter 1

Introduction

In broadest generality, a situation awareness network or gossip network can best be described as a network of nodes that can sense, transmit, relay and receive information. The type of information can range from weather measurements to road traffic conditions to enemy military vehicles. In such a network, each node wants a most updated view of all the available information. Deriving a policy on whether to sense, transmit, relay or receive is one of the main topics of study.

Situation awareness networks are becoming more prevalent in the research fields and in practice. A large part of this field is the study of road traffic networks, where cars transmit, relay and receive information about congestion or blockage. Information propagation in these networks is studied extensively. A measure of performance for such networks is the time until all nodes are fully aware of all information. These networks are usually studied through the use of simulation and deriving conjectures from these simulations that can later be proven.

This report focuses on an analytic approach of situation awareness networks. We consider static communicating nodes and are interested in the age of information one node has about another node. A general, abstract model is introduced where the type of information is undefined. The model can later be specialized towards one type of network, this is beyond the scope of this study. The main goal is to design an algorithm capable of iterating a joint distribution of ages of information along two branches of nodes from a source to a sink. Using this algorithm, marginal and joint distributions of ages of information with respect to a chosen source can be studied.

In the literature we find that queuing networks with catastrophes, such as [2], are related to the age of information in situation awareness networks. A study on gossip networks in [1] focuses on propagation of information in networks with mobile nodes. Another link can be made to PERT, a project planning analysis, which is a type of shortest (or critical) path problem.

Articles such as [3], [4] and [5] discuss PERT problems with dependent activities and means of solving these. These different fields are all related to our general model and can be used, yet differ enough for this to be a unique study.

We commence with an introduction of the general model in Chapter 2, the model will be made more specific by introducing a policy on the reception and transmission behaviour in one of the later sections. Chapters 3 and 4 discuss two special cases of the general model, for both cases an algorithm is sought. The penultimate chapter, Chapter 5, discusses the design of an algorithm that can be used in iterating over two branches of nodes. The report is concluded in the final chapter.

Chapter 2

A general model

In this chapter we introduce the general model of the situation awareness network. More structure is given to this model throughout Sections 2.1 and 2.2 by introducing assumptions and policies on transmission and reception behaviour. The final sections investigate the effects of the proposed assumptions.

Consider a network of N nodes, $\mathcal{N} = \{1, \ldots, N\}$. Each node performs sensing, wireless communication and uses the information sensed by the other nodes of the network. Each node attempts to maintain an updated situation awareness view of the information sensed by all other nodes in the network, yet it may be that the information is not always up to date. Nodes communicate by broadcasting. When a node broadcasts, it sends a packet that contains a subset of its situation awareness view. This may contain the information sensed by the node, as well as information taken from its situation awareness view regarding other nodes at earlier times. Thus nodes essentially relay sensor information. Note that we assume that if node i broadcast its situation awareness view about node j, it sent all of the information regarding that node as well as a time stamp of the age of that information. Nodes can not receive information when they are broadcasting themselves, analogous to a transceiver.

We assume the network evolves in discrete (slotted) time $n = 0, 1, 2, \ldots$. The situation awareness age process is $\{X_n(i,j), n = 0, 1, \ldots, (i,j) \in \mathcal{N}^2\}$. This is essentially the state of our system. The element $X_n(i,j)$ is the age of the information that node i has about node j at time n. Thus for example if $X_n(1,3) = 15$, we know that at time n, node 1's most updated view regarding the sensed information at node 3 is from time $n - 15$.

Given initial conditions, $X_0(\cdot)$, the evolution of $X_n(\cdot)$ is driven by the following objects: the binary matrix sequence of information transmissions $\{I_n(i,j), n = 0, 1, \ldots, (i,j) \in \mathcal{N}^2\}$, the binary vector sequence of information sensing $\{F_n(i), n = 0, 1, \ldots, i \in \mathcal{N}\}$ and the sequence of channel

functions $\{C_n, n = 0, 1, \ldots\}$ where $C_n : \{0,1\}^N \to \{0,1\}^{N\times N}$. For the information transmission sequence, we have that $I_n(i,j) = 1$ if and only if at time n node i has broadcasted its information regarding node j. For the sensing sequence, we have that $F_n(i) = 1$ if and only if at time n node i has updated its own sensor information (by sensing). For the operation of the channel functions, we use an additional variable, $T_n(i) = \mathbf{1}\{\sum_{j=1}^N I_n(i,j) \geq 1\}$, where $\mathbf{1}\{\}$ is the indicator function. This sequence determines if node i broadcasted a packet at time n (note that all sensor information that i broadcasts is assumed to sit on one packet). Now the function $C_n(T_n)$ results in the matrix with elements $R_n(i,j)$ where $R_n(i,j) = 1$ if and only if j received a packet sent by node i. The diagonal elements $R_n(i,i)$ are meaningless and set to 0. Note that we restrict the sequence C_n to result in $R_n(i,j) = 1$ only if $T_n(i) = 1$.

Given the above primitives, the evolution of the situation awareness age is as follows:

$$X_{n+1}(i,j) = \begin{cases} \left(X_n(i,j) \wedge \bigwedge_{k:R_n(k,i)I_n(k,j)=1} X_n(k,j)\right) + 1 & i \neq j, \\ (1 - F_n(i))(X_n(i,i) + 1) & i = j. \end{cases} \tag{2.1}$$

The symbol $\bigwedge_i$ is used to indicate taking a minimum over i. Observe that the variables that drive the recursion are the initial conditions, X_0, the broadcasts I_n, the sensing actions F_n and the channel conditions C_n. A minimum is taken over the set of information pieces regarding node j that have been received by node i. Each node is only interested in the youngest information and therefore compares the minimum age of information that was received with the current age of information stored in node i. The channel plays a role here in transforming I_n first to T_n (putting information pieces in transmitted packets) and then through C_n (which models packet losses) to R_n. Further note that if we ignore the sensing actions by assuming that each node performs sensing at every time instant, we have that $X_n(i,i) = 0$.

The above model is very general and without probabilistic assumptions. We narrow down, but still remain fairly general. First we ignore the degree of freedom of sensing and assume $F_n(i) \equiv 1$, a node will always have perfect information about itself and therefore $X_n(i,i) = 0$ as stated earlier. In the next sections Bernoulli channels and policies are introduced. These govern the transmission and reception behaviour.

2.1 Bernoulli channels

An approximation of real-life communication channels is made by introducing a Bernoulli variable governing reception of a signal. We shall assume the channel functions, C_n are an i.i.d. sequence with the following structure:

for all $i, j \in \mathcal{N}$ and $A \subset \mathcal{N}$, let $p_{ij}(A)$ indicate the probability of successful reception at node j of packet transmitted at node i when the set of transmitting stations is A, where $A = \{k : T_n(k) = 1\}$. Require that $p_{ij}(A) = 0$ if $i \notin A$. Now assume that the i, j'th entry of $C_n(T_n)$ is a bernoulli random variable taking 1 with probability $p_{ij}(A)$ and 0 otherwise, independent of all other random variables. The probability $p_{ij}(A)$ is a function of the transmitting nodes to incorporate for interference and for the fact that we deal with nodes that can not receive and transmit at the same time.

It is sensible to assume that if $i \in A, B$ and $A \subset B$ then $p_{ij}(A) \geq p_{ij}(B)$. Thus $p_{ij}(\{i\})$ is the success probability on $i \to j$ without any possible interference from any other node. We denote this probability as p_{ij} for short.

We impose a graph structure between the nodes by specifying a set of directed links. We can then define the set of neighbours of node j as K_j. The nodes in the set K_j are the nodes that can influence the reception probability of node j, i.e. nodes that are close by or nodes that have a powerful signal, disrupting other communication over large distances. We can model interference of other broadcasting nodes by a decreasing interference function L, depending on the transmitting nodes that affect the receiving node. Other broadcasting nodes, which are not neighbours, do not influence the reception probabilities. The proposed success probability is defined as

$$p_{ij}(A) = \mathbf{1}\{j \notin A\} L(A \cap K_j) p_{ij}. \tag{2.2}$$

Note that $p_{ij}(A) = p_{ij}(A \cap K_j)$ which is desired behaviour, and we cannot receive if we are transmitting. A choice can be made for the interference function L, one can think of mutual exclusiveness, which can be described as, if some neighbouring node of node j is transmitting, node j cannot receive. Another option is a decreasing function over the number of transmitting neighbouring nodes, e.g. $L = \frac{1}{|A \cap K_j| + 1}$. One can also study the behaviour when there is no interference by setting L to 1.

2.2 Bernoulli policies

The transmission control of the nodes in the network is governed by $I_n(i, j)$. Recall that $I_n(i, j) = 1$ if and only if at time n node i has broadcasted its information regarding node j. A policy on the transmission probabilities is an engineering decision, not a model of nature. One could opt for a different policy that is not treated in the report. We shall now describe policies where I_n is an i.i.d. sequence, independent of the state and refer to these as Bernoulli policies. In greatest generality we have a probability distribution function over $\mathcal{N}^2$ indicating the probability of transmission of each subset. A way to define this is to assume that $I_n(i, j)$ equals 1 with probability q_{ij} and 0 otherwise, independent of all other i and j. This implies that each node

i transmits a random number of information updates, which can also be 0, because $q_i = 1 - \prod_j (1 - q_{ij}) \leq 1$. An alternative way is to assume that each node will transmit all bits of information with probability q_i (independently of all other nodes). In this report we solely focus on the first option and look at just one type of information being sent or relayed, i.e. j is a node that does not change, labeled as the *source*.

2.3 Reception probabilities induced by Bernoulli channels and policies

In the previous two sections the Bernoulli channels and policies were introduced. These channels and policies induce a successful reception probability. Let $\{\gamma_j(B) : B \subset \mathcal{N}\}$ be the one-step probability distribution indicating from which sensors reception will occur at node j, where B is the set of nodes from which the successful reception occurs. We can write this in terms of the given parameters as

$$\gamma_j(B) = \sum_{D:B\subset D\subset\mathcal{N}} \prod_{i\in\mathcal{N}\setminus\mathcal{D}} (1-q_i) \prod_{i\in D} q_i \prod_{i\in B} p_{ij}(D) \prod_{i\in D\setminus B} (1-p_{ij}(D)). \tag{2.3}$$

One can edit equation (2.3) by interchanging q_i for q_{ik} for a specific node k to obtain the probabilities of receiving a specific packet of information (about node k). This is useful in assessing the probabilities of updates on $X_n(j,k)$. Let us also define, for $B \subset C \subset \mathcal{N}$

$$\gamma_j(B;C) = \sum_{D:D\subset N\setminus C} \gamma_j(B\cup D).$$

This is the modified probability of reception γ_j, given that we only take the nodes in set C into account, i.e. we look at a subset of the whole network and want to know the probabilities of reception.

Equation (2.3) can also be altered to obtain the probability of successful reception of one specific type of information throughout the whole network. Let $\{\lambda_j(B) : B \subset \mathcal{N}\}$ be the one-step probability distribution over all subsets of $\mathcal{N}$ indicating which sensor receives information about source j. Specifically, this can only be used in networks that have a tree-structure. In such networks, each node has only one possible incoming transmission but is allowed to transmit to multiple nodes.

$$\lambda_j(B) = \sum_{D:A\subset D\subset\mathcal{N}} \prod_{i\in\mathcal{N}\setminus\mathcal{D}} (1-q_{ij}) \prod_{i\in D} q_{ij} \prod_{i\in B} p_i(D) \prod_{i\in D\setminus B} (1-p_i(D)) \tag{2.4}$$

Note that there is only a subtle difference between γ_j and λ_j. We have introduced the probability $p_i(A)$ which denotes the probability of reception

at node i given that nodes A are transmitting. It is defined analogous to (2.2), where K_j is now defined as the set of neighbours that could cause interference at node j and p_{ij} is interchanged for p_i. Recall that there is only one possible incoming transmission at node i. The variable λ_j will be used to great extent in analyzing networks that have a tree-structure. Once again, let us also define, for $B \subset C \subset \mathcal{N}$

$$\lambda_j(B;C) = \sum_{D:D\subset N\setminus C} \lambda_j(B \cup D). \tag{2.5}$$

Throughout the rest of this report, tree structures will be studied to great extent and therefore the variable $\lambda_j(B)$ is used in numerous cases. The M-file used to compute these probabilities of successful reception is shown in Appendix B.

2.4 Implication of the transceiver and interference principles on a homogeneous system

The general model introduced the principle of a transceiver, nodes cannot receive information when they are transmitting information themselves. One could argue that there should be some optimal value for probability of transmission q_i. If we set q_i to either 0 or 1 for all i, there will not be reception on any of the nodes. We illustrate this optimal value by treating an example.

We model a homogenous system without interference. Each node in the system has a transmission probability q that is varied over the range $[0, 1]$. We specify a network with a tree-structure consisting of two branches with 4 links on each branch, contributing to a total of 8 links. Each node only transmits to one node, except for the source, it transmits to the first node on both branches. The term link refers to the directed transmission of information from one node to the next. The reception probability of each link is labeled $p_{ij}(\{i\}) = p$ for all i and j in the example model. The following is obtained after running the M-file for computing $\lambda_j(B)$ and we plot the probability of have no reception on any of the nodes $\lambda(\emptyset)$, Figure 2.1.

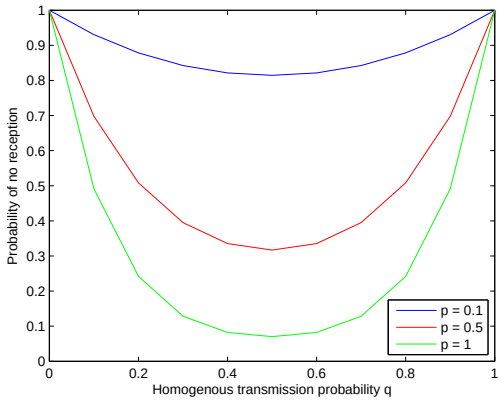

Figure 2.1: Effect of the transmission probability on the probability of no reception **without** interference

We can conclude that in a homogenous system without interference, the value that minimizes the probability of no reception is 0.5 and one can clearly identify a parabola over the domain. The ideal ratio between receiving and transmitting is 1 if we want to minimize $\lambda(\emptyset)$.

We can study the other side of the spectrum as well, a homogenous system with interference. We model the interference function as $L = \frac{1}{|A \cap K_j| + 1}$ and declare for each node j that the set K_j consists of all nodes that are not

directly communicating to node j. Figure 2.2 shows that for such a 'chaotic' interference scheme, the optimal q value is a lot lower than the case without interference.

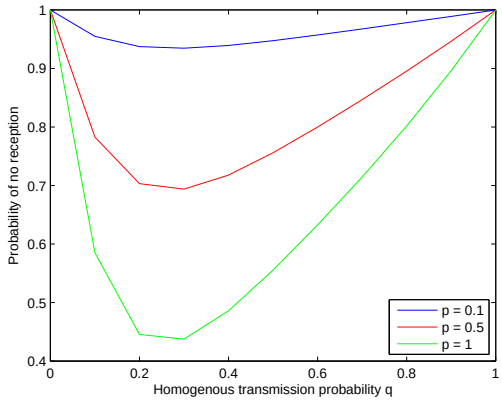

Figure 2.2: Effect of the transmission probability on the probability of no reception **with** interference

Whilst this section does not give any proof or theorem on optimal q values, it does give insight into how the transceiver and interference principles affect the probability of reception. One could derive that if there is a lot of interference, somewhat lower q values achieve better information propagation. Another simple, yet crucial observation is that for interference heavy networks, the probability of no reception rises, in accordance with what would be expected.

Chapter summary

In this chapter we introduced the general model and Bernoulli channels and policies without losing too much generality. An example was treated to show the effects of the transceiver and interference in the general model. In the next chapter we will look at a special case of the general network, the single hop networks.

Chapter 3

Single hop networks

A single hop network is a network consisting of an arbitrary number of sets of transmitter and receiver, where we do not look at the transmission behaviour of the receiver (without loss of generality we can view the receiver as a sink).

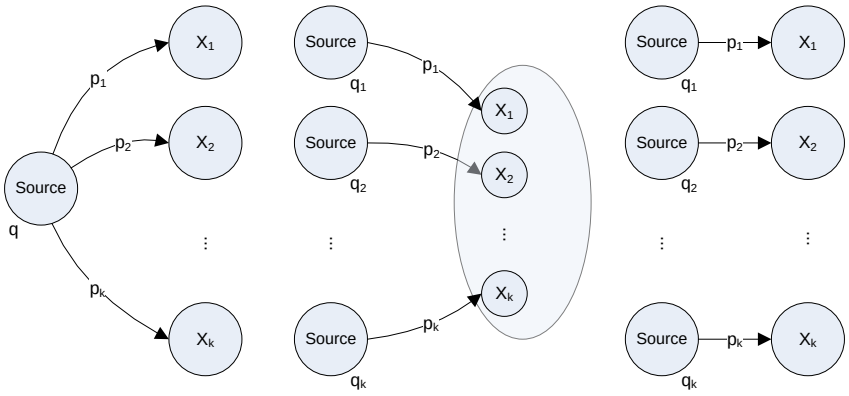

Figure 3.1: Three extreme types of single hop networks

We can identify three extreme types. The first identifiable type is a network where a number of receiving nodes share a common source or transmitter. This type is shown on the left in Figure 3.1. The second type is shown in the middle of the figure. Here we have sources transmitting to the same receiver. The last type is distinguished by the fact that no nodes are shared. Due to the layout of the first network type, the age of information processes $X_1, X_2, \ldots, X_k$ are dependent as long as the transmitter does not always transmit ($q < 1$). The other two types can have dependence: interference can still occur and cause dependence between the age of information processes. The processes $X_1, X_2, \ldots, X_k$ all give the age of information of their linked source, the subscript n has been omitted for clarity. The three types

of single hop networks share the same underlying Markov chains with the same transitions, this allows us to group these types and derive a way to analyze them. One can also think of combinations of the three presented types, which will still give us the same structure.

On a side note, the underlying Markov chain is related to a queueing network with arrivals to a number of workstations with a probability of a catastrophe job arriving to a queue, 'killing' all the jobs in the line. The catastrophe in a single hop network occurs when a packet of information is successfully received at a receiver and it updates its age of information to 0, thus 'emptying the queue'.

In this chapter we start of by deriving analytic results for single hop networks with 1, 2 and 3 age of information processes. For the two single hops we derive the equilibrium distribution and validate it by showing that the marginal distribution information is still contained within the joint distribution. An expression for the covariance is derived for the two single hops case and it is shown how interference affects the joint distribution. Also, we are interested in what reception probabilities allow for the joint distribution to be the product of the two marginal distributions. We study the single hop network with 3 age of information processes and continue to extend the search for joint distributions of single hop networks to an arbitrary number of links. The chapter is concluded with some remarks on how to compute the minimum of a discrete joint distribution and an illustrative example.

3.1 One single hop

We look at the simplest model of a transmitter (node 1) and a receiver (node 2), modeled by two nodes. Each timestep, the first node transmits with probability q and the second node will receive this with probability p, due to imperfect channel conditions. The situation awareness age process is $\{X_n(2,1), n = 0, 1, \ldots\}$. Recall that $X_n(2,1)$ is the age of information that node 2 has about node 1 at time n. Normally, the minimum value of $X_n(2,1)$ is 1, equal to the minimum age of the previous node (in this case, 0) plus 1. We subtract the value 1 from $X_n(2,1)$ to obtain a neater model without changing the system in any way. The state space of this Markov chain is represented in Figure 3.2.

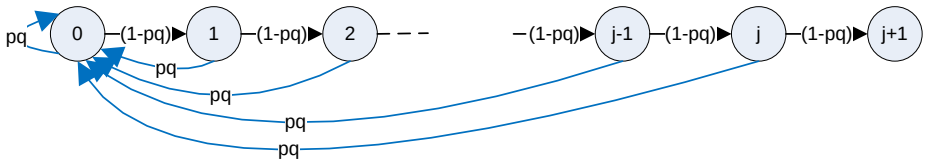

Figure 3.2: State space of the Markov chain of one single hop.

This Markov chain is irreducible, aperiodic and recurrent. We are interested in the equilibrium distribution of this Markov chain. We define the probability that we are in state i (in equilibrium) as

$$\pi_i = \lim_{n\to\infty} \mathbb{P}(X_n(2,1) = i).$$

The equilibrium equations can be derived from the above figure to obtain

$$\pi_0 = pq\pi_0 + pq\pi_1 + \ldots = pq\sum_{i=0}^{\infty} \pi_i, \tag{3.1}$$
$$\pi_i = (1-pq)\pi_{i-1} = (1-pq)^2\pi_{i-2} = \ldots = (1-pq)^i\pi_0, \quad i \geq 1.$$

Which are subject to the normalization equation $\sum_i \pi_i = 1$, due to the fact that the probabilities have to sum up to 1. We can immediately see from (3.1) and the normalization equation that $\pi_0 = pq$ and therefore the equilibrium distribution of the model is a geometric distribution with parameter $(1-pq)$ and is given by

$$\pi_i = (1-pq)^i pq, \quad i \geq 0.$$

3.2 Two single hops

In this section we consider a model of two single hops, consisting of three nodes. We identify two different age of information processes $\{X_n, n = 0, 1, \ldots\}$, and $\{Y_n, n = 0, 1, \ldots\}$. We are interested in the joint equilibrium distribution of these two processes and we will verify that the marginal distributions are geometric, in accordance with the distribution of one single hop. We will show that interference and covariance are related and this affects the joint distribution.

The marginal distribution of one single hop is related to the two single hops network. Once the marginal distributions of the two nodes with the correct parameters are obtained, dependence does not affect these distributions, it only affects the joint distribution.

We denote four reception probabilities, $\lambda(\emptyset)$ is the probability of no reception, $\lambda(\{1\})$, $\lambda(\{2\})$ and $\lambda(\{1,2\})$ denote the probabilities of reception only at X_n, only at Y_n or at both, respectively. These four probabilities sum to 1. Note that this model can include interference by allowing that $(\lambda(\{1\}) + \lambda(\{1,2\}))(\lambda(\{2\}) + \lambda(\{1,2\})) \neq \lambda(\{1,2\})$. The joint distribution we are looking for also possesses that property, it will be correct for both dependent and independent systems. Once again, we subtracted the value 1 from both age of information processes. The state space of this model and its transitions are shown in Figure 3.3.

Equilibrium distribution

We are interested in the equilibrium distribution of this Markov chain. We define the probability that we are in state (i, j) (in equilibrium) as

$$\pi_{i,j} = \lim_{n\to\infty} \mathbb{P}(X_n = i, Y_n = j).$$

The equilibrium equations were derived as

$$\begin{aligned}
\pi_{0,0} &= \lambda(\{1,2\}) \sum_{i,j} \pi_{i,j}, \qquad (3.2)\\
\pi_{i,0} &= \lambda(\{2\}) \sum_{j=0}^{\infty} \pi_{(i-1),j} \quad , \ i \geq 1,\\
\pi_{0,j} &= \lambda(\{1\}) \sum_{i=0}^{\infty} \pi_{i,(j-1)} \quad , \ j \geq 1,\\
\pi_{i,j} &= \lambda(\emptyset) \cdot \pi_{(i-1),(j-1)} \quad , \ i, j \geq 1.
\end{aligned}$$

The first three equilibrium equations are boundary equations. The normalization equation is given as $\sum_{i,j} \pi_{i,j} = 1$, in combination with (3.2) we obtain that $\pi_{0,0} = \lambda(\{1,2\})$ in the same manner as in the previous model.

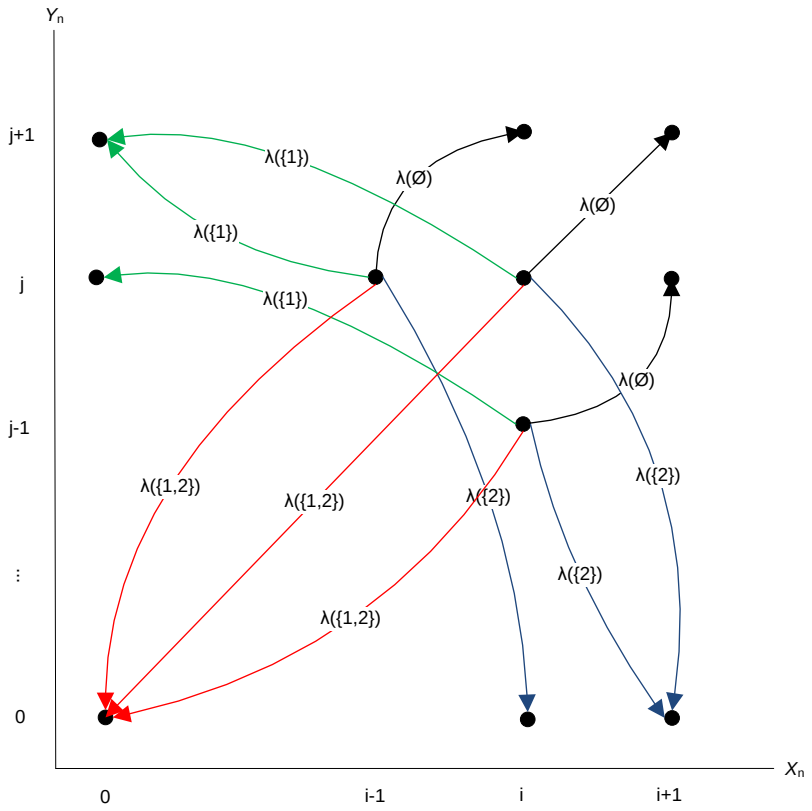

Figure 3.3: 2-Dimensional state space of the Markov chain of two single hops.

In the second and third equation we sum over one entire coordinate, therefore we can use the marginal distribution obtained in the previous section, given as

$$\begin{aligned}
\pi_i^{(X_n)} &= \sum_{j=0}^{\infty} \pi_{i,j} = \lim_{n\to\infty} \sum_{j=0}^{\infty} \mathbb{P}(X_n = i, Y_n = j) = \lim_{n\to\infty} \mathbb{P}(X_n = i) \\
&= (1 - (\lambda(\{1\}) + \lambda(\{1,2\})))^i (\lambda(\{1\}) + \lambda(\{1,2\})), \\
\pi_j^{(Y_n)} &= \sum_{i=0}^{\infty} \pi_{i,j} = \lim_{n\to\infty} \sum_{i=0}^{\infty} \mathbb{P}(X_n = i, Y_n = j) = \lim_{n\to\infty} \mathbb{P}(Y_n = j) \\
&= (1 - (\lambda(\{2\}) + \lambda(\{1,2\})))^j (\lambda(\{2\}) + \lambda(\{1,2\})).
\end{aligned}$$

We denote a marginal distribution by $\pi_i^{(.)}$, i.e. the marginal distribution of parameters $(.)$ at the value i. First we will introduce two new variables to increase readability. These govern the behaviour of the joint and marginal

distributions

$$c_1 = 1 - (\lambda(\{2\}) + \lambda(\{1,2\})),$$
$$c_2 = 1 - (\lambda(\{1\}) + \lambda(\{1,2\})).$$

The physical interpretation of these variables is that c_1 the probability of having no reception on the second node, and c_2 is the probability of having no reception on the first node. Using these two new variables and the marginal distributions, the equilibrium equations simplify to

$$\begin{aligned}
\pi_{0,0} &= \lambda(\{1,2\}), \\
\pi_{i,0} &= \lambda(\{2\})\pi_{i-1}^{(X_n)} = \lambda(\{2\})c_2^{i-1}(1-c_2) \quad , i \geq 1, \\
\pi_{0,j} &= \lambda(\{1\})\pi_{j-1}^{(Y_n)} = \lambda(\{1\})c_1^{j-1}(1-c_1) \quad , j \geq 1, \\
\pi_{i,j} &= \lambda(\emptyset) \cdot \pi_{(i-1),(j-1)} \quad , i,j \geq 1.
\end{aligned}$$

This can be rewritten to obtain a general geometric formula over the diagonals

$$\pi_{i,j} = \begin{cases} \lambda(\emptyset)^i \lambda(\{1,2\}) & \text{for } i = j, \\ \lambda(\emptyset)^j \lambda(\{2\}) c_2^{i-j-1}(1-c_2) & \text{for } i > j, \\ \lambda(\emptyset)^i \lambda(\{1\}) c_1^{j-i-1}(1-c_1) & \text{for } i < j. \end{cases} \tag{3.3}$$

Marginal distribution from the joint distribution

As a sanity check, we will show that the joint distribution indeed possesses the two marginal distributions we used earlier. We can obtain the marginal of X_n by summing over the Y_n coordinate, namely j. The state space along this line is partitioned into three sets, as the expression for $\pi_{i,j}$ also consists of three parts.

$$\begin{aligned}
\pi_i^{(X_n)} &= \sum_{j=0}^{\infty} \pi_{i,j} = \sum_{j=0}^{i-1} \pi_{i,j} + \pi_{i,i} + \sum_{j=i+1}^{\infty} \pi_{i,j} \\
&= \lambda(\{2\})(1-c_2) \sum_{j=0}^{i-1} c_2^{i-1} \left(\frac{\lambda(\emptyset)}{c_2} \right)^j + \lambda(\emptyset)^i \lambda(\{1,2\}) \\
&\quad + \lambda(\emptyset)^i \lambda(\{1\}) \sum_{j=i+1}^{\infty} (1-c_1) c_1^{-(i+1)} c_1^j \\
&= (1-c_2)(c_2^i - \lambda(\emptyset)^i) + \lambda(\emptyset)^i \lambda(\{1,2\}) + \lambda(\emptyset)^i \lambda(\{1\}) = (1-c_2)c_2^i.
\end{aligned}$$

The marginal distribution of X_n is, as expected, a geometric distribution with parameter c_2. Note that no assumptions were made on interference or dependence, X_n and Y_n could be dependent, yet still the same geometric marginal distribution is obtained. Similar to the derivation of the marginal distribution for X_n, we can derive the expression for Y_n and find that is geometric with parameter c_1.

Interference and covariance

A measure of the dependence is the covariance between the two variables X and Y (the subscript n is omitted for clarity). It is defined as

$$\begin{aligned}\mathrm{Cov}(X,Y) &= E((X-E(X))(Y-E(Y))) \\ &= E(XY - XE(Y) - YE(X) + E(X)E(Y)) \\ &= E(XY) - E(X)E(Y). \end{aligned} \tag{3.4}$$

Independent variables X and Y ensure the covariance is zero. However, a covariance of zero does not mean that the two variables are independent, it is a one-way relation. The expected value of the joint and the marginal distributions are needed to calculate the covariance. See Appendix A for the derivations. The covariance is symmetric and can then be expressed as

$$\mathrm{Cov}(X,Y) = \frac{\lambda(\emptyset)\lambda(\{1,2\}) - \lambda(\{1\})\lambda(\{2\})}{(\lambda(\{1\}) + \lambda(\{1,2\}))(\lambda(\{2\}) + \lambda(\{1,2\}))(1-\lambda(\emptyset))}.$$

If there is no interference between the two transmitters, the covariance has to be equal to 0 and the joint distribution is the product of the two marginal distributions, $\pi_{i,j} = \pi_i^{(X)}\pi_j^{(Y)}$. For no interference $(\lambda(\{1\})+\lambda(\{1,2\}))(\lambda(\{2\})+\lambda(\{1,2\})) = \lambda(\{1,2\})$ has to hold, meaning that the probability of reception of both is the product of the (independent) probability that one of them receives. As this is a quadratic equation, there are two possible values for $\lambda(\{1,2\})$ that satisfy this equation. These two values that satisfy the previous equation, are also the only two values for $\lambda(\{1,2\})$ that satisfy $\mathrm{Cov}(X,Y) = 0$. We deduced that in the independent case one of the two gave an infeasible result and thus we can conclude that the value

$$\begin{aligned}\lambda(\{1,2\}) = \frac{1}{2}\Bigg(\Big(\big(\lambda(\{1\}) - \lambda(\{2\})\big)^2 - 2\lambda(\{1\}) - 2\lambda(\{2\}) + 1\Big)^{\frac{1}{2}} \\ + (1 - \lambda(\{1\}) - \lambda(\{2\}))\Bigg)\end{aligned} \tag{3.5}$$

is the parameter value that sets the covariance equal to 0 and the joint distribution is the product of the two marginal distributions. The infeasible result for $\lambda(\{1,2\})$ indeed does not give a joint distribution that is the product of the two marginal distributions. We can conclude that if and only if (3.5) holds, the joint distribution is the product of the two marginal distributions.

3.3 Three single hops

In this section we will briefly show that the explicit results obtained in the two single hops case can also be extended to a three dimensional Markov chain of the same structure. We consider a model with three age of information processes, structured as three single hops. These processes are labeled X_1, X_2 and X_3. We denote by $\lambda(A)$ the probability of reception on the nodes in A, where A can be any set of the powerset of $\mathcal{N} = \{1, 2, 3\}$. The equilibrium equations are given and we argue that these can be written in an explicit form.

Equilibrium distribution

We are interested in the equilibrium distribution of this Markov chain. The probability that we are in state (i, j, k) in equilibrium is defined as

$$\pi_{i,j,k} = \lim_{n\to\infty} \mathbb{P}(X_1 = i, X_2 = j, X_3 = k)$$

where the subscript n of each age of information process is omitted. Once again, the value 1 is subtracted from the processes. The equilibrium equations were derived as

$$\begin{aligned}
\pi_{0,0,0} &= \lambda(\{1,2,3\}) \sum_{i,j,k} \pi_{i,j,k} = \lambda(\{1,2,3\}), \\
\pi_{i,0,0} &= \lambda(\{2,3\}) \sum_{j,k} \pi_{i-1,j,k} = \lambda(\{2,3\}) \pi_{i-1}^{(X_1)} \quad , i \geq 1, \\
\pi_{0,j,0} &= \lambda(\{1,3\}) \sum_{i,k} \pi_{i,j-1,k} = \lambda(\{1,3\}) \pi_{j-1}^{(X_2)} \quad , j \geq 1, \\
\pi_{0,0,k} &= \lambda(\{1,2\}) \sum_{i,j} \pi_{i,j,k-1} = \lambda(\{1,2\}) \pi_{k-1}^{(X_3)} \quad , k \geq 1, \\
\pi_{i,j,0} &= \lambda(\{3\}) \sum_{k} \pi_{i-1,j-1,k} = \lambda(\{3\}) \pi_{i-1,j-1}^{(X_1,X_2)} \quad , i,j \geq 1, \\
\pi_{i,0,k} &= \lambda(\{2\}) \sum_{j} \pi_{i-1,j,k-1} = \lambda(\{2\}) \pi_{i-1,k-1}^{(X_1,X_3)} \quad , i,k \geq 1, \\
\pi_{0,j,k} &= \lambda(\{1\}) \sum_{i} \pi_{i,j-1,k-1} = \lambda(\{1\}) \pi_{j-1,k-1}^{(X_2,X_3)} \quad , j,k \geq 1, \\
\pi_{i,j,k} &= \lambda(\emptyset) \pi_{i-1,j-1,k-1} \quad , i,j,k \geq 1.
\end{aligned}$$

As one can see, the 3-dimensional joint distribution depends on known 1- and 2-dimensional marginal distributions of combinations of the nodes. Note that all edges and faces of the cube are calculated explicitly. The interior of the cube can then be calculated by using the last equilibrium equation. The structure of these equilibrium equations is in essence the same as the ones

obtained from the two single hops case. In that 2-dimensional Markov chain the edges are known explicitly and the interior is computed from these edges. Thus, using the same methodology a general geometric expression over the diagonals can be obtained. One can split the state space into 13 subsets, in the same manner the formula for $\pi_{i,j}$ was split into three parts. We refrain from deriving these subsets and their corresponding explicit expressions as there is little gain in showing them after it was derived that such explicit formulaes exist.

3.4 Arbitrary number of single hops

We now look at a network of N sensors that transmit over one-hop to a sink, equivalent to a network of N single hops, where the set of nodes is $\mathcal{N}$. Bernoulli policies and reception probabilities are used, which allow for interference to be modeled. Recall that expressions (2.4) and (2.5) are used to compute reception probabilities for all sets of the powerset of all nodes. The latter equation is used when we are looking at a subset of the complete network and do not want to take reception into account on the nodes that are not in de subset. We allow C to be a proper subset of $\mathcal{N}$ in order to consider smaller networks in the recursive specification that follows. We specify the N-dimensional Markov chain $X = (X_1, X_2, \ldots, X_N)$ and state that all reception probabilities p_i and transmission probability q_i are strictly positive and therefore X is a irreducible, non-periodic Markov chain over the state-space $\mathbb{Z}_+^N$. Establishing positive-recurrence from the equilibrium equations is obvious, as there is a positive probability to have a *catastrophe* and return to the value 0 on one of the X_i. Note that the value 1 is subtracted from all ages without losing any information. The analysis of the stationary distribution of X is constructed through marginal distributions of increasing dimension.

Let $C = \{i_1, i_2, \ldots, 1_{|C|}\} \subset \mathcal{N}$ and the ages of information of each node in the set C are ordered $0 \leq x_{i_1} \leq \ldots \leq x_{i_{|C|}}$. Then,

$$\lim_{n\to\infty} \mathbb{P}(X_{i_1} = x_{i_1}, \ldots, X_{i_{|C|}} = x_{i_{|C|}}) = \pi^{(C)}_{x_{i_1},\ldots,x_{i_{|C|}}}$$

is constructed recursively as denote in algorithm 1.

ALGORITHM 1 - Joint distribution of $|C|$ single hops.

0. Set $k = \min\{j : x_{i_j} \leq x_{i_{j+1}}\}$, taking $k = |C|$ if $x_{i_1} = \ldots = x_{i_{|C|}}$.
1. if $x_{i_{|C|}} = 0$, set

$$\pi^{(C)}_{0,\ldots,0} = \lambda(C; C) \tag{3.6}$$

elseif $x_{i_{|C|}} > 0$, set

$$\pi^{(C)}_{x_{i_1},\ldots,x_{i_{|C|}}} = \begin{cases} \pi^{(C)}_{0,\ldots,0,x_{i_{k+1}}-x_{i_1},\ldots,x_{i_{|C|}}-x_{i_1}} \cdot \lambda(\emptyset; C)^{x_{i_1}} & x_{i_1} > 0 \\ \pi^{(C\setminus\{i_{k+1},\ldots,i_{|C|}\})}_{x_{i_{k+1}}-1,\ldots,x_{i_{|C|}}-1} \cdot \lambda(C\setminus\{i_{k+1},\ldots,i_{|C|}\}; C) & x_{i_1} = 0 \end{cases} \tag{3.7}$$

If we want to compute the joint distribution of the nodes in C, we can find this distribution from the above algorithm. From the 2- and 3-dimensional case we know that the origin is always equal to the probability of reception on all nodes in the network, this is also the case in this algorithm, see (3.6). If we are in the interior of the state space, thus the smallest age is strictly positive, $x_{i_1} > 0$, we can compute this probability by moving back along to the diagonal to the nearest (hyper) plane or edge and using the knowledge that there is a geometric decay along the diagonals. This is shown in the first part of equation (3.7). If we are already on a (hyper) plane or line, we can use the marginal distribution of all the other nodes that have a strictly positive age. This was derived from the equilibrium equations of the 2- and 3-dimensional single hop Markov chains. As an example, we show that for $N = 1$ the algorithm derives the following equations

$$\pi_{x_1}^{(X_1)} = \lambda(\{1\};\{1\}) \cdot \lambda(\emptyset;\{1\})^{x_1}$$

which is equal to what was derived in Section 3.1. For $N = 2$ the algorithm derives the following equations

$$\pi_{x_1,x_2}^{(X_1,X_2)} = \begin{cases} \pi_{x_2-1}^{(X_2)} \cdot \lambda(\{1\};\{1,2\}) & 0 = x_1 < x_2, \\ \pi_{0,x_2-x_1}^{(X_1,X_2)} \lambda(\emptyset;\{1,2\}) & 0 < x_1 < x_2, \\ \lambda(\{1,2\};\{1,2\}) \cdot \lambda(\emptyset;\{1,2\})^{x_1} & x_1 = x_2. \end{cases}$$

The above equation only covers half of the state space, but it can be noted that when interchanging X_1 and X_2 the results can be obtained for the other half of the plane. Note that these are the same equilibrium equations as the ones obtained from the 2-dimensional case.

3.5 Minimum over a joint distribution of a single hop network

Once one has obtained the joint distribution of the nodes that are within a sink, e.g. think of the second type of single hop network in Figure 3.1, where all sources share the same sensing information from a sensor, the minimum has to be taken to get the marginal distribution of the sink. This phenomenon is explained by the fact that a node always wants the most updated information and if it receives from two previous at the same time, it will only take the most up-to-date information. The minimum in a discrete state space can be computed by setting one of the ages to a specific value and summing over all larger values in the other dimensions. Setting an age to a specific value should be done for all dimensions and the minimum for that specific value is then equal to the summation of the computed sums. One should note not to count any state more than once. An example is treated to illustrate the above explanation.

We consider the two single hops network of Section 3.2. Recall that there are two age of information processes X_n and Y_n (with the value 1 subtracted from their ages), if we consider these to be part of the sink we can take the minimum to obtain the marginal distribution, let us define the corresponding stochastic variable as

$$Z_n = \min(X_n, Y_n).$$

We are interested in $\pi_z^{(Z)} = \lim_{n\to\infty} \mathbb{P}(Z_n = z)$, the equilibrium distribution of Z_n. The minimum over the 2-dimensional Markov chain can be computed by summing over 2 lines.

The computation of the equilibrium distribution can be expressed as

$$\begin{aligned}\pi_z^{(Z)} &= \lim_{n\to\infty} \Bigg(\sum_{k=1}^{\infty} \mathbb{P}(X_n = z, Y_n = z+k) \\ &\quad + \sum_{k=1}^{\infty} \mathbb{P}(X_n = z+k, Y_n = z) + \mathbb{P}(X_n = z, Y_n = z) \Bigg) \\ &= \sum_{k=1}^{\infty} \pi_{z,z+k} + \sum_{k=1}^{\infty} \pi_{z+k,z} + \pi_{z,z}.\end{aligned}$$

If we now use the derived expressions for $\pi_{i,j}$, the above computation can

be expressed as

$$\begin{aligned}\pi_z^{(Z)} &= \lambda(\emptyset)^z \lambda(\{1\})(1-c_1)\sum_{k=0}^{\infty} c_1^k + \lambda(\emptyset)^z \lambda(\{2\})(1-c_2)\sum_{k=0}^{\infty} c_2^k \\ &\quad + \lambda(\emptyset)^z \lambda(\{1,2\}) \\ &= \lambda(\emptyset)^z \left(\lambda(\{1\}) + \lambda(\{2\}) + \lambda(\{1,2\})\right) \\ &= \lambda(\emptyset)^z (1-\lambda(\emptyset)).\end{aligned}$$

We can conclude that for two geometric stochastic variables and any dependence structure, the minimum of these two variables also has a geometric distribution. The interference can also be found in the above expression, as $\lambda(\emptyset)$ is larger when there is interference and more probability mass is present at higher levels.

The same methodology can be applied to an N-dimensional Markov chains as long as their joint distribution is known.

Chapter summary

In this chapter we studied the single hop networks and finally we were able to derive an algorithm that computes the joint distribution of an arbitrary number of single hops. The 2-dimensional case was analyzed in greater detail and it was found that the joint distribution is the product of its marginal distribution for one value of $\lambda(\{1,2\})$. The chapter was concluded with an explanation on how to obtain the marginal distribution of a sink. In the next chapter a line of relays is studied and an algorithm is derived that iteratively computes joint distributions of two succeeding nodes.

Chapter 4

Line of relays

We now consider a model in which nodes can only transmit information to the next node. We are then able to follow the age of information with respect to a given source. Such a model of N nodes is named a line of relays, as $N-2$ nodes relay the information from the first node to the last node. We are interested in the joint distribution of two last nodes in the line and the marginal distributions of each node along the line. Some of this information is used when computing a marginal distribution for a node where two or more of these branches come together and we have to take a minimum to obtain its marginal distribution. We will now study a line of relays such as shown in Figure 4.1 and present an algorithm to iteratively calculate joint distributions of two succeeding nodes and all marginal distributions.

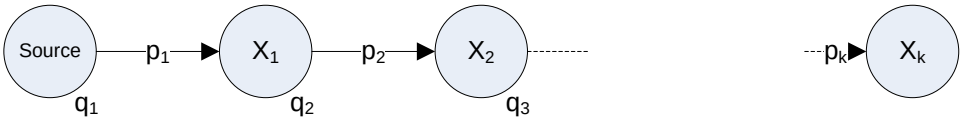

Figure 4.1: Line of relays

It is important to note that the age of information of X_i depends on X_{i-1}, as $x_{i-1} \leq x_i$, where we labeled the current age x. This means that the joint distribution (X_i, X_{i+1}) depends on the joint distribution (X_{i-1}, X_i). If we receive information from X_{i-1} we will use the conditional stationary distribution

$$\pi_{x_{i-1}|x_i}^{(X_{i-1}|X_i)} = \lim_{n\to\infty} \mathbb{P}(X_{i-1} = x_{i-1}|X_i = x_i) = \frac{\pi_{x_{i-1},x_i}^{(X_{i-1},X_i)}}{\pi_{x_i}^{(X_i)}}$$

for the current value x_i and all possible values x_{i-1} to modify the transition probabilities, recall that the subscript n was omitted.

Let $\lambda(\{1\})$ be the probability of a successful reception at node X_i, $\lambda(\{2\})$ the probability of a successful reception at node X_{i+1}, $\lambda(\{1,2\})$ at both and

$\lambda(\emptyset)$ the probability that no node receives. There are no absorbing states in this 2-dimensional Markov chain and thus the probabilities of the outgoing transitions should sum to 1. This is still the case, as $\sum_{i=0}^{j} \pi_{i|j}^{(X_{i-1}|X_i)} = 1$. Figure 4.2 displays the transition probabilities in the state space of two succeeding nodes. Superscripts of the conditional stationary distributions were omitted. The blue contour indicates that each node within that contour has the same structure of outgoing transitions.

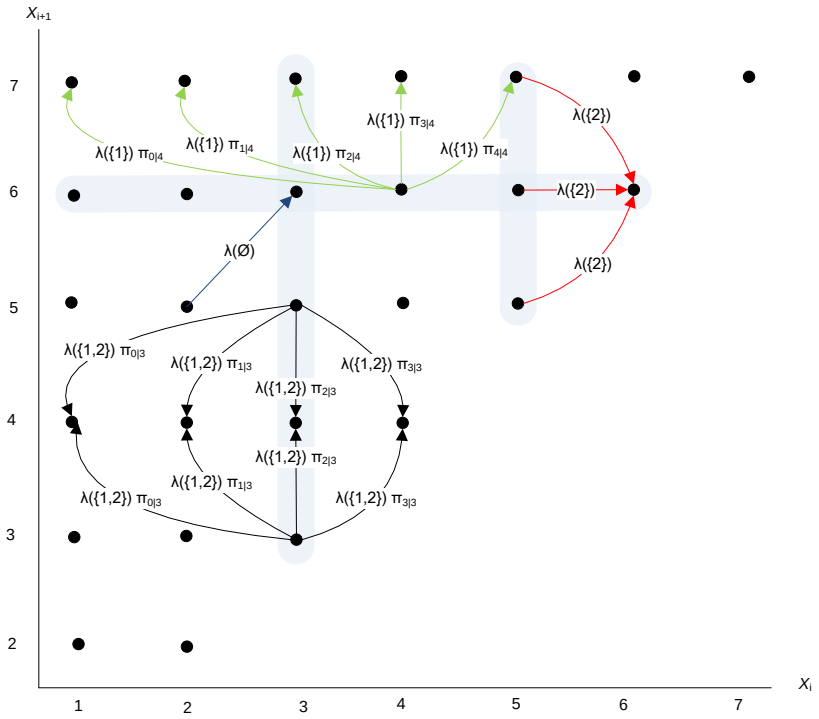

Figure 4.2: State space and transition probabilities of two succeeding nodes in a line of relays

Equilibrium distribution

From the transition diagram we are able to derive the equilibrium equations. The equilibrium equations are used in the algorithm to obtain an exact joint distribution up to a bounding box of arbitrary size M. We split the state space into five parts, the two points at the bottom, the two edges and the interior points. If there is no superscript, the current joint distribution (X_i, X_{i+1}) is intended.

$$
\begin{aligned}
\pi_{1,2} &= \lambda(\{1,2\})\pi_{0,1}^{(X_{i-1},X_i)}, \\
\pi_{2,2} &= \lambda(\{2\})\pi_1^{(X_i)} + \lambda(\{1,2\})\pi_{1,1}^{(X_{i-1},X_i)}, \\
\pi_{1,j} &= \lambda(\{1\}) \sum_{k=1}^{j-1} \pi_{0|k}^{(X_{i-1}|X_i)} \pi_{k,j-1} + \lambda(\{1,2\})\pi_{0,j-1}^{(X_{i-1},X_i)} \quad , j \geq 3, \\
\pi_{i,i} &= (\lambda(\emptyset) + \lambda(\{1\})\pi_{i-1|i-1}^{(X_{i-1}|X_i)})\pi_{i-1,i-1} + \lambda(\{2\})\pi_{i-1}^{(X_i)} \\
&\quad + \lambda(\{1,2\})\pi_{i-1,i-1}^{(X_{i-1},X_i)} \quad , i \geq 3, \\
\pi_{i,j} &= \lambda(\emptyset)\pi_{i-1,j-1} + \lambda(\{1\}) \sum_{k=i-1}^{j-1} \pi_{i-1|k}^{(X_{i-1}|X_i)} \pi_{k,j-1} \\
&\quad + \lambda(\{1,2\})\pi_{i-1,j-1}^{(X_{i-1},X_i)} \quad , j > i, i \geq 2.
\end{aligned}
$$

Now that the equilibrium equations are known, we can iteratively compute joint distributions with the use of algorithm 2.

ALGORITHM 2 - Joint distribution of two succeeding nodes in a line of relays.

0. Use the known $\pi_{i,j}^{(X_{i-1},X_i)}$ and $\pi_i^{(X_i)}$ and set $\pi_{1,2}$ and $\pi_{2,2}$.

1. Iterate until a bounding box of size M is reached.

for $j = 3 : M$
  for $i = 1 : j$
    Calculate $\pi_{i,j}$ using the above equilibrium equations
  end
end

The number of operations required by this algorithm is roughly $\frac{M^2}{2}$. In section 3.1 we showed that the marginal distribution of the first node of the chain is geometric. We also know that the source always has perfect information and thus the joint distribution of the source and the first node

is exactly the geometric distribution along the axis where the age of information of the source is 0. Supplying this joint and marginal distribution to the algorithm allows us to calculate all joint distributions of two succeeding nodes of a line of relays.

A result of this algorithm is that we are able to obtain the marginal distribution of each node. We see that the marginal distribution of X_i is a generalized negative binomial with success probabilities $\lambda(\{1\};\{1\})$, $\lambda(\{2\};\{2\})$, ..., $\lambda(\{i\};\{i\})$ in the notation of (2.5). A generalized negative binomial is a special type of a discrete phase-type distribution. The phase-type distribution is the distribution obtained from a terminating Markov chain with transition probability matrix

$$P = \begin{bmatrix} T & (I-T)\mathbf{T^0} \\ \mathbf{0} & 1 \end{bmatrix}$$

where T is a block matrix containing the transition probabilities of the transient states and $\mathbf{T^0} = (I-T)\mathbf{e}$ and $\mathbf{e}$ is a row vector of ones of corresponding size. In our case, the matrix T is defined as

$$T = \begin{bmatrix} 1-\lambda(\{1\};\{1\}) & \lambda(\{1\};\{1\}) & 0 & \cdots & \\ & \ddots & \ddots & & \\ 0 & & & \ddots & \lambda(\{i-1\};\{i-1\}) \\ & & & 0 & 1-\lambda(\{i\};\{i\}) \end{bmatrix}.$$

The probability mass function of the phase type distribution is given as

$$f(k) = \boldsymbol{\alpha} T^{k-1}\mathbf{T^0} \quad , k = i, i+1, \ldots$$

and $\boldsymbol{\alpha}$ is the initial distribution, for our model this is $\begin{bmatrix} 1 & 0 & 0 & \cdots \end{bmatrix}$. We start at $k = i$ because this is the least number of jumps you need to be able to have k successes, i.e. the information about the source reaches X_i in a minimum of i timesteps.

This can be interpreted as the marginal being a sum of all geometric distributions of the previous nodes, where the parameter of the j'th geometric distribution is $\lambda(\{j\};\{j\})$. This is a clean result, allowing us to know all marginal distributions in a network that has a tree-structure. Simulations were ran for tree structures and the same results were obtained.

Chapter summary

In this chapter we derived an algorithm to iteratively compute joint distributions of two succeeding nodes in a line. As a result, we are able to derive the marginal distribution of each node. These marginal distributions are generalized negative binomial distributions. So, for networks with a tree-structure all single marginal distributions, i.e. of one node, are known. In

the next chapter we will analyze a network with two branches, where we use the knowledge obtained in the previous chapters.

Chapter 5

A network with two branches

In the previous two chapters we derived the steady state distributions of single hop networks and the joint distribution of two nodes in a line of relays. We also looked at ways to compute the minimum over a joint distribution of two nodes. This knowledge is used in obtaining the distribution of the age of information at a sink given there are two routes from the source to the sink. Note that we deal with a tree structure right up until the point we reach the sink. Dependence between the links occurs due to the fact that the two branches are connected at the source, as we have already discovered in the single hops network. Interference can still occur between the two branches or from other nodes in the surrounding network. Figure 5.1 shows the model we are describing. The motivation for looking at such a model is the analogy with an intelligent transport system in a city grid, where we want to send information from one intersection to the next intersection through the two shortest routes.

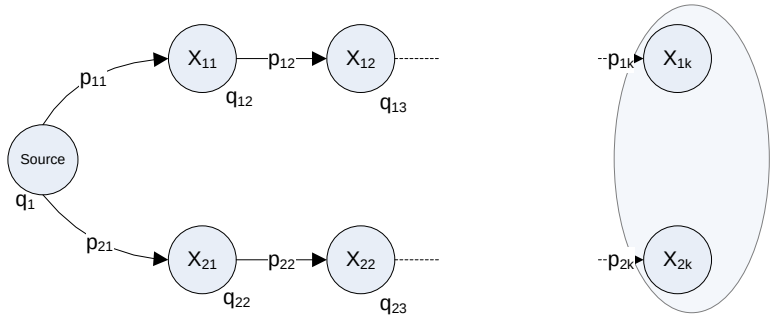

Figure 5.1: Two branches of a tree-structured network

The main goal is to be able to iterate over these two branches by continually calculating a four dimensional joint distribution, starting at the source. For this iteration we first need to gain insight in the three- and four-dimensional joint distributions. The first section of this chapter discusses the algorithm

and equilibrium equations used to calculate the three-dimensional distribution. We continue by showing the four-dimensional case and finally we present the main iteration algorithm.

5.1 The 3-dimensional case

The model we are looking at now is shown below in Figure 5.2. There are two nodes on the first branch and one on the second. It is a matter of notation where we put the second node, it could be placed on the second branch as well.

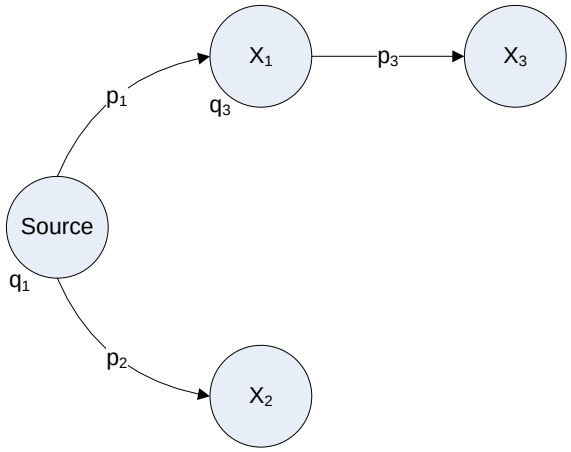

Figure 5.2: One source and two branches with 2 nodes on the first branch and 1 on the second

The layout of the model dictates that the state space is

$$\{(x_1, x_2, x_3) : 0 \leq x_1 \leq x_3, 0 \leq x_2, 1 \leq x_3\},$$

and we have dependence between X_1 and X_2 because they are connected at the source and there is dependence between X_1 and X_3 as they are two subsequent nodes in a line. The state space of this model can be visualised by taking the state space of the two succeeding relays, which is a plane, shown in Figure 4.1 and moving this in the third dimension and relabeling the axes. We continue by describing the equilibrium equations in the next subsection.

Equilibrium equations

Below we present the equilibrium equations for the 3-dimensional Markov chain. Note that the value 1 has been subtracted from all ages, such that the minimum of the first two nodes is 0.

$$
\begin{aligned}
\pi_{0,0,1} &= \lambda(\{1,2,3\}) \sum_{j,k} \pi_{0,j,k} = \lambda(\{1,2,3\}) \pi_0^{(X_1)}, \\
\pi_{1,0,1} &= \lambda(\{2,3\}) \sum_{j,k} \pi_{0,j,k} = \lambda(\{2,3\}) \pi_0^{(X_1)}, \\
\pi_{i,0,i} &= \lambda(\{2,3\}) \sum_{j,k} \pi_{i-1,j,k} + \lambda(\{2\}) \sum_{j} \pi_{i-1,j,i-1} \quad , \ i \geq 2 \\
&= \lambda(\{2,3\}) \pi_{i-1}^{(X_1)} + \lambda(\{2\}) \pi_{i-1,i-1}^{(X_1,X_3)}, \\
\pi_{0,0,k} &= \lambda(\{1,2\}) \sum_{i,j} \pi_{i,j,k-1} + \lambda(\{1,2,3\}) \sum_{j,l} \pi_{k-1,j,l} \quad , \ k \geq 2 \\
&= \lambda(\{1,2\}) \pi_{k-1}^{(X_3)} + \lambda(\{1,2,3\}) \pi_{k-1}^{(X_1)}, \\
\pi_{i,0,k} &= \lambda(\{2\}) \sum_{j} \pi_{i-1,j,k-1} = \lambda(\{2\}) \pi_{i-1,k-1}^{(X_1,X_3)} \quad , \ i \geq 1, k \geq 2, i > k, \\
\pi_{0,j,1} &= \lambda(\{1,3\}) \sum_{i,k} \pi_{i,j-1,k} = \lambda(\{1,3\}) \pi_{j-1}^{(X_2)} \quad , \ j \geq 1, \\
\pi_{0,j,k} &= \lambda(\{1\}) \sum_{i} \pi_{i,j-1,k-1} = \lambda(\{1\}) \pi_{j-1,k-1}^{(X_2,X_3)} \quad , \ j \geq 1, k \geq 2, \\
\pi_{1,j,1} &= \lambda(\{3\}) \sum_{k} \pi_{0,j-1,k} = \lambda(\{3\}) \pi_{0,j-1}^{(X_1,X_2)} \quad , \ j \geq 1, \\
\pi_{i,j,i} &= \lambda(\{3\}) \sum_{k} \pi_{i-1,j-1,k} + \lambda(\emptyset) \pi_{i-1,j-1,i-1} \quad , \ i \geq 2, j \geq 1 \\
&= \lambda(\{3\}) \pi_{i-1,j-1}^{(X_1,X_2)} + \lambda(\emptyset) \pi_{i-1,j-1,i-1}, \\
\pi_{i,j,k} &= \lambda(\emptyset) \pi_{i-1,j-1,k-1} \quad , \ i,j \geq 1, k \geq 2, i,j > k.
\end{aligned}
$$

The only marginal distribution that is not known is $\pi_{j,k}^{(X_2,X_3)}$ and we can therefore not explicitly find the equilibrium distribution. We will use a similar approach as the one used in algorithm 2. This approach is summarized below in algorithm 3. The structure of the equilibrium equations allows us to consequently compute the equilibrium probabilities of each plane (X_1, X_3), in the exact same way as was denote in algorithm 2. The algorithm works by computing the probabilities $\pi_{i,j,k}$ for the first plane, where $x_2 = 0$, and using this iterate for the next plane, where we continue to use the previous plane to compute the probabilities of the current plane, until a bounding cube of size M is filled.

ALGORITHM 3 - Joint distribution of a network with 2 branches, 2 nodes on the first branch and 1 on the second.

0. Obtain the marginal and joint distributions $\pi_{i,j}^{(X_1,X_2)}, \pi_{i,k}^{(X_1,X_3)}, \pi_i^{(X_1)}, \pi_j^{(X_2)}$ and $\pi_k^{(X_3)}$ from earlier results or using algorithm 2 and set $\pi_{0,0,1}$ and $\pi_{1,0,1}$.

1. Find $\pi_{i,0,k}$ in the same fashion as algorithm 2, until a bounding box of size M is reached.

for $k = 2 : M$
  for $i = 0 : k$
    Calculate $\pi_{i,0,k}$
  end
end

2. Iterate this plane by increasing j by one each step, until a cube of size M is filled.

for $j = 1 : M$
  Set $\pi_{0,j,1}$ and $\pi_{1,j,1}$
  for $k = 2 : M$
    for $i = 0 : k$
      Calculate $\pi_{i,j,k}$
    end
  end
end

This algorithm gives us the exact values for $\pi_{i,j,k}$ up until a bounding cube of size M. We can validate the joint distribution by comparing it to the joint distribution of two succeeding nodes in a line by summing out the single node on the second branch. Another check that is performed, is obtaining the marginal for one node from the joint distribution and checking this against the generalized negative binomial expression. Both validation checks hold and the above algorithm has been shown to work correctly.

5.2 The 4-dimensional case

The final case we will be studying is a 4-dimensional Markov chain, shown in Figure 5.3. The obtained joint distribution will be the main iterate used in the iteration scheme, denoted in the next section.

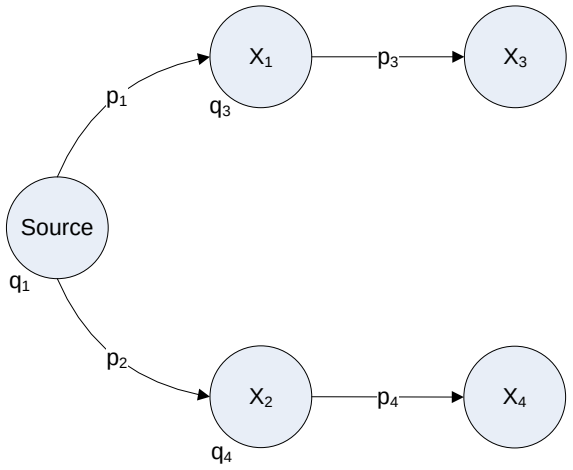

Figure 5.3: One source and two branches with 2 nodes on both branches

Once again, the layout of the model dictates that the state space is

$$\{(x_1, x_2, x_3, x_4) : 0 \leq x_1 \leq x_3, 0 \leq x_2 \leq x_4, 1 \leq x_3, 1 \leq x_4\},$$

where the value 1 is subtracted from the ages. One can imagine the state space in the following way, take the 3-dimensional state space of the previous model and set the maximum value of x_2 to the current value of x_4. If we increment x_4 by one, the next 3-dimensional state space contains one more plane than the previous. We would then obtain a number of 3-dimensional state spaces which together make the 4-dimensional state space.

The equilibrium equations are build analogously to the ones shown in the subsection of the 3-dimensional model. However, there are 25 different regions that we can identify and showing the equilibrium equations is overly tedious. The structure of the equations allows us to first compute the first triangular prism for $x_4 = 1$ in almost the same way as in the 3-dimensional case. The computed equilibrium probabilities are used to calculate the next ones by incrementing x_4. The algorithm to compute the joint distribution works in the same way as algorithm 3, where the outer loop is over the x_4 variable and the x_2 variable loops from 0 until x_4.

The results of this algorithm are compared to the 3-dimensional joint distribution by summing over either x_3 or x_4. The same validation checks as for the 3-dimensional model were done as well. The results match and we

conclude that this algorithm is successful. In the next section we will use the obtained knowledge on 3- and 4-dimensional joint distributions, together with the conditioning used for the two succeeding relays, to iterate the latter distribution over a tree network with two branches.

5.3 Iterating over two branches

Using the algorithms shown in the previous two sections, one can compute the 3- and 4-dimensional joint distributions when the first two nodes are connected to a source. We have seen how insight in the structure of the equilibrium equations can be used to compute the equilibrium distribution exactly. We would now like to be able to iterate this 4-dimensional joint distribution over the two branches, such that the number of required operations is linear in the number of succeeding 4-dimensional distributions. The number of computations for one distribution is in the order of M^4. From the line of relays we have learned that one can compute the joint distribution of two succeeding nodes by using the conditional distribution of the previous two succeeding nodes. Whenever there is successful reception on the first node, we condition on the probability of the previous node having a certain value given that we know the value of the first node, and alter the probabilities of jumping to a state accordingly. The same alteration has to be made to be able to compute the joint distribution of four nodes when they are not connected to a source. The idea behind the iteration scheme is shown below in Figure 5.4, where we are interested in the distribution of the nodes in the green contour and the joint distribution of the nodes in red is used as the conditional distribution. Looking at memory, the algorithm has to store the previous joint distribution to compute the current joint distribution and any intermediate arrays used in the computation of the current one.

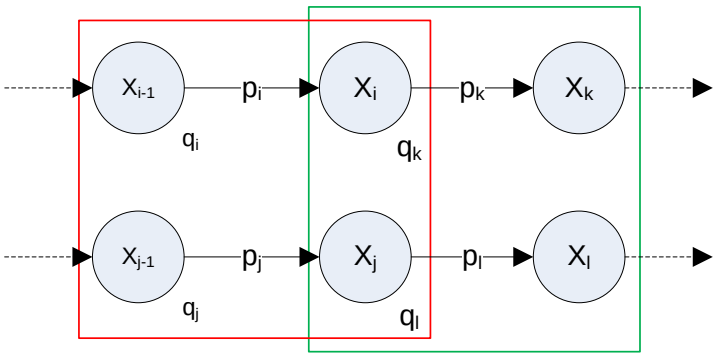

Figure 5.4: Setup of the iteration scheme used to subsequently compute four dimensional joint distributions

When using the conditioning for successful reception in nodes i and j, we introduce a large number of new transitions, especially for $\lambda(\{i,j\})$. These transitions depend on the previous 4-dimensional conditional distribution, which is in this case $\pi_{i-1,j-1|i,j}^{(X_{i-1},X_{j-1}|X_i,X_j)}$. Due to the conditioning, most of the transitions in the equilibrium equations do not sum to a marginal distribution anymore, as they are multiplied by different factors from the con-

ditional distribution. Therefore, equilibrium equations usually depend on a large subset of the state space and there is no neat structure that can be exploited with a previously obtained algorithm such as algorithm 3. Most of the points in the state space have incoming transitions from states very far away, which cannot be summed to a marginal distribution. Truncation has to be used at a level M to be able to get a joint distribution. Then we are able to construct a system of linear equations that is of the form

$$Ax = b$$

where x is a lexicographical ordering of the state,

$$x = [\pi_{0,0,1,1}, \pi_{0,1,1,1}, \ldots, \pi_{0,M,1,1}, \pi_{1,0,1,1}, \ldots, \pi_{M,M,1,1}, \pi_{0,0,2,1}, \ldots, \\ \pi_{M,M,M,1}, \pi_{0,0,1,2}, \ldots].$$

Each row of the matrix A is an equilibrium equation, note that some rows explicitly state that a value $\pi_{i,j,k,l} = 0$, as they are outside the feasible state space domain. One would expect the vector b only has zeros, as we are dealing with a Markov chain. This is not the case for this system. If we have reception on nodes k and/or l, we can sum out this variable and we are left with a marginal distribution of nodes i, j and possibly node k or l that did not receive a packet. These marginal distributions can be computed beforehand and supplied to the algorithm that constructs the A and b matrices, such that they end up on the right hand side of the equation. Hence, the vector b has non-zero elements and using normalization is not necessary; increasing the size of the bounding box will ensure the summation over the equilibrium probabilities converges to 1, and thus the approximate solution converges to the exact solution. The M-file that computes each iterate can be found in Appendix C and the main M-file for running the iteration scheme can be found in Appendix D. In the next subsection we address memory and time constraint problems we encountered, the subsection after that validates the proposed iteration scheme and M-file. The final subsection concludes with some remarks on the length of the two branches.

Memory and time constraints

In the iteration scheme we are dealing with a joint distribution of length M in all four dimensions. This leads to an A matrix of size M^4-by-M^4. For a moderate value for M, a full matrix would lead to out of memory errors on a 2 GB RAM system. We therefore need to gain insight into how Matlab handles memory and study the algorithm, as initially, the computation of one iterate would take 18 hours on an Intel Dual Core T7200 2.00 GHz laptop. We use [6] as a guide to the optimization.

Matlab uses *copy on write*, which means that in an assignment of one variable to the other, it only passes a pointer as long as the data is unaltered.

Once we change the contents, it copies the array to a different memory location and implements the changes, hence the name *copy on write*. In most newer versions of Matlab the *in-place computation* functionality has been added. In an assignment $A = A + 1$, Matlab does not have to copy the matrix A to a new memory location and add the value 1 to each entry, it notices we are using the same memory address and it will append the matrix without having to copy it. This is especially useful when working with large matrices which would flood the memory when they need to be copied.

Matlab is column oriented and it therefore prefers to store elements column by column, or use vectors instead of row vectors. For small arrays no real difference can be noted, in our case where we are constructing the A matrix a significant difference can be seen when storing elements row or column wise.

Matlab prefers matrix and vector multiplication over loops, hence the name Matrix Laboratory. As of version 6.5 (2002), MathWorks introduced a Just-In-Time (JIT) Accelerator that accelerates loops by an enormous amount, decreasing computation time by a factor 80. Still, Matlab prefers matrix multiplication over loops with JIT acceleration by a fair margin.

In a coding situation that cannot be optimized any further, keep in mind that the Matlab language is intended primarily for easy prototyping rather than high-speed computation. In some cases, an appropriate solution is Matlab Executable (MEX) external interface functions. With a C or Fortran compiler, it is possible to produce a MEX function that can be called from within Matlab in the same way as an M-function. The typical speed improvement over equivalent M-code is easily ten-fold.

The profiler tool in Matlab allows us to identify bottlenecks in the code, after running the code it shows a report of what lines where called, the number of times these lines were called and where the most time was spent. This is an extremely useful feature when trying to optimize an M-file. While using this tool, a lot was learned about what functions are slow when pushing memory usage and computation time to its limits. Some of the main findings are discussed below.

Do not concatenate. Matlab has to find a new continuous block of memory of a larger size and it has to copy the contents. Preallocating vectors and matrices can be used to avoid this problem. If the final size of the matrix is unknown, one can run the code a few times and get an upper bound on the dimensions, which can then be used in the preallocation. If this is not an option, there are user-supplied M-files that handle memory management in a more efficient way than Matlab, adding a predefined size of memory to the current memory address when the vector or array is nearly filled.

Do not predefine a sparse matrix and index into a sparse matrix afterwards, this is computationally expensive. Instead, use three vectors that describe

the row, column and value of each element in the sparse matrix and use these three vectors to build the sparse matrix afterwards.

Some built-in functions can be slow, these functions have to perform checks and should work for many more different types of input than the single one we are supplying to the function. Inlining a function such that it does not have to perform any checks and specifically writing it for one purpose is a good way to save computation time. As an example, the *find* function was used on a vector of size M^4, by instead using logical indexing on a vector $ind = 1 : M^4$ a simpler approach was introduced and the computation time was reduced by a factor 10. The methodology of writing own versions of functions and reducing time was used for numerous other built-in functions as well. In one part of the code, some transpose of a 4-dimensional array was needed, this was solved by using the built-in function *permute*. It appeared to be slow, so, when constructing this 4-dimensional array, indexing into the matrix was swapped and the array was already transposed when outputted to the main function again.

The only remaining bottleneck is the logical indexing of a large vector, which can not easily be modified in a computationally attractive way. The logical indexing is called in the order of M^4 times and takes up about 91% computation time of the algorithm. When trying to improve the code, one could look into using a MEX function.

After having studied the memory management and using the profiler extensively, the computation time has been cut by more than a factor 30 and we can handle systems for which the bounding box is of size $M = 27$, all on the same system as described before. Using a better system and the newest version of Matlab allows us to increase the size of the bounding box. For now, this maximum size works well and it captures almost all of the equilibrium probabilities.

Validation of the iteration scheme

The iteration scheme does not give us an exact 4-dimensional distribution because of the truncation that has to be used. A validation of the model is therefore even more important. For increasing M, more equilibrium probabilities are computed on the outskirts of the state space and all the equilibrium probabilities will become more accurate, due to a higher truncation level. For the validation, we use a model with 3 nodes on both branches, this allows us to compute two 4-dimensional iterates. The first iterate is the joint distribution of the first two nodes on both branches and the second iterate is the joint distribution of the last two nodes on both branches. Some random values are taken for the $\lambda(A)$ probabilities, such that they sum up to 1.

A measure for the accuracy of the joint distribution is the summation of the distribution, once it gets closer to 1 it is a better approximation. Figure 5.5 shows the summation over the two iterates. Note that even the exact distribution would not sum to 1 for low levels of M, as there is probability mass on the outside of the bounding box as well. We are able to compute the first iterate for higher values of M as the first two nodes are connected to a source and using the conditional distribution does not introduce new transitions, i.e. this gives the exact distribution. For large M, the summation of the first iterate is equal to 1.

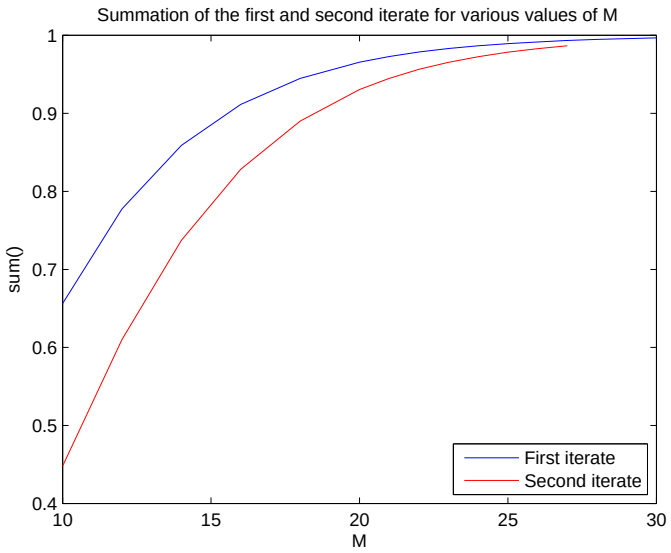

Figure 5.5: Summation of the first and second iterate, closer to 1 means a better approximation

The second validation we use is a comparison with the exact values that were

obtained from the knowledge that the marginal distribution is a generalized negative binomial distribution. We use a system with the maximum value for M. Node 3 is the second node on the first branch and node 5 is the third node on the first branch. Figures 5.6 and 5.7 show this comparison. For node 3, both iterates can be used, for node 5 only the last iterate can be used. What is good to note, is that the exact solution is an upper bound to the obtained marginal distribution from the iterates and the one from the first iterate is always above the one from the second iterate. All of these remarks are as expected. As no validation check is unsuccessful, it can be concluded that the algorithm is indeed correct and can be used to iterate the 4-dimensional joint distribution from a source to a sink over 2 branches.

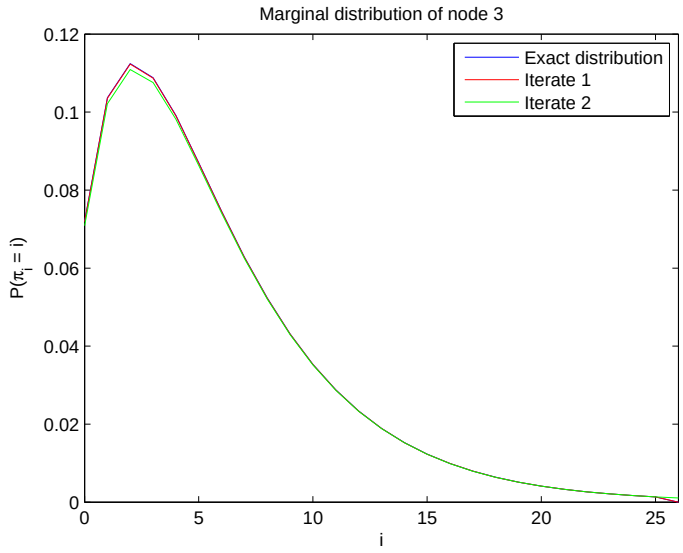

Figure 5.6: Exact marginal distribution of the third node compared with the ones obtained from the first and second iterate

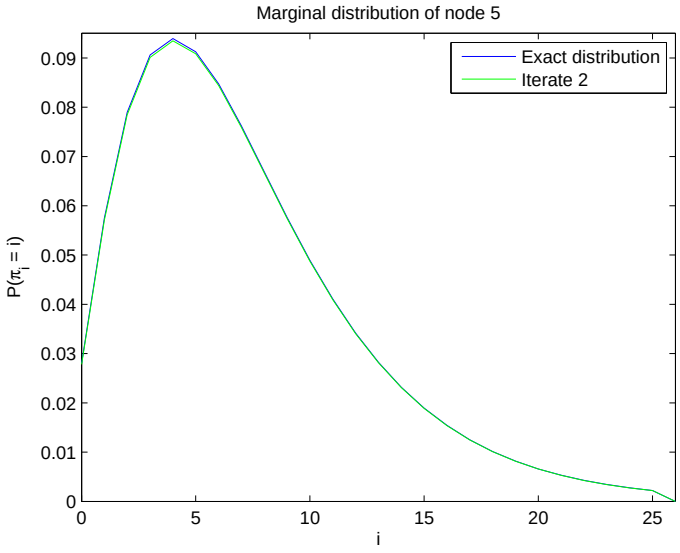

Figure 5.7: Exact marginal distribution of the third node compared with the one obtained from the second iterate

Branches of different length

The algorithm needs a previous 4-dimensional iterate to be able to compute the current joint distribution. In the current scheme, we are only able to handle branches that have the same length, where the length is expressed in the number of hops. If we reach the last hop on one branch, we intuitively shift up a node on the second branch and keep the same two nodes on the first branch, creating a diamond shape contour if we use the methodology presented in Figure 5.4. This intuition is incorrect, as the algorithm now demands a different, unknown, 4-dimensional distribution. An easy way to solve this problem is by introducing dummy nodes on the shorter branch such that the length of both branches are the same. These dummy nodes always receive from the node before and therefore have the same age of information distribution as the node before, shifted by the value 1, as we moved up by 1 hop. The iteration scheme can then be used with the same rectangular contour discussed before. To correct for the extra hops that were introduced, one has to shift one, or possibly two, of the axes. Keep in mind that this technique does not give the correct 4-dimensional distribution as the first node is usually also a dummy node and not the correct one but last node on the first branch. However, it is still useful as we are interested in the joint distribution of the last node on both branches to obtain the distribution of the sink.

Chapter summary

This chapter discussed a network consisting of two paths leading from source to the sink. We were interested in the marginal distribution of the sink. An algorithm was derived and validated to iteratively compute joint distributions of four dimensions, such that the number of operations is linear in the number joint distributions. Problems concerning memory and time were tackled by gaining insight into Matlab routines and memory management. We have validated the iteration scheme and the obtained results are as expected. Finally, it was shown that the algorithm is able to derive the marginal distribution for branches of different length.

Chapter 6

Conclusion

In this report we have introduced an abstract model of situation awareness networks and have specifically limited us to Bernoulli channel and reception policies. Even though this is a simplification of policies used in practice, the performance analysis done still gives great insights into possible solving methods. Using this model, the age of information was studied in great detail.

We have distinguished between two types of models. The first type are the single hop networks, consisting of sets of two nodes. By using the knowledge obtained from small numbers of single hops, an algorithm was derived that can compute the joint distribution of an arbitrary number of single hops. The second type is the line of relays. By studying this network, it was found that the marginal distribution of a node in a tree-structured network is a generalized negative binomial. An algorithm capable of finding joint distributions of sets of succeeding nodes in these lines of relays was introduced and validated as well.

Finally, a network consisting of two branches from source to sink was studied. We have succeeding in deriving an algorithm that computes joint distributions of four dimensions, such that the number of operations is linear in the number of joint distributions. Each iterate we move one hop closer (on each branch) to the sink. Problems concerning memory and time were tackled by gaining insight into Matlab routines and memory management. The algorithm is also capable of handling branches of different length by introducing dummy links, without changing the joint distribution of the last two nodes, needed for the marginal distribution of the sink. This iteration scheme was validated with the help of the generalized negative binomial expressions and the already validated algorithms of the single hop networks and the line of relays.

A first step has been set in the field of analysis of age of information in situation awareness networks. For future work, one should improve the

algorithm to be able to deal with an arbitrary number of paths from source to sink. Ultimately, a scheme should be used that can obtain mean ages of information and marginal distribution for every layout of the network. If one succeeds herein, the next thing to do is tackle real-time control problems concerning the transmission probabilities q where the age of information is minimized. Another option would be to study different, more complex policies or even moving nodes such as in road traffic situation awareness networks. The analysis could be coupled with more extensive simulations to give evidence for the algorithms.

Bibliography

[1] Augustin Chaintreau, Jean-Yves Le Boudec, Nikodin Ristanovic *The Age of Gossip: Spatial Mean Field Regime.* SIGMETRICS '09, 2009, pp. 109-120.

[2] Xiuli Chao *A queueing network model with catastrophes and product form solution.* Operations Research Letters 18, 1995, pp. 75-79.

[3] Gideon Weiss *Stochastic bounds on distributions of optimal value functions with applications to pert, network flows and reliability.* Annals of Operations Research 1, 1984, pp. 59-65.

[4] Pierpaolo Pontrandolfo *Project duration in stochastic networks by the PERT-path technique.* International Journal of Project Management 18, 2000, pp. 215-222.

[5] Arunava Banerjee, Anand Paul *On path correlation and PERT bias.* European Journal of Operational Research 189, 2008, pp. 1208-1216

[6] Pascal Getreuer *Writing fast MATLAB code.* MathWorks, 2009

Appendix A

Derivation of the covariance for a network with two single hops

In this appendix the covariance of zero mean of the joint distribution is obtained, secondly the expected value of the marginal distributions of a model with two single hops are derived, these are used in computing the covariance between the two single hops. Recall that the joint distribution is

$$\pi_{i,j} = \begin{cases} \lambda(\emptyset)^i \lambda(\{1,2\}) & \text{for } i = j \\ \lambda(\emptyset)^j \lambda(\{2\}) c_2^{i-j-1}(1-c_2) & \text{for } i > j \\ \lambda(\emptyset)^i \lambda(\{1\}) c_1^{j-i-1}(1-c_1) & \text{for } i < j \end{cases}$$

Where we have introduced two variables for readability, as

$$c_1 = (1 - (\lambda(\{2\}) + \lambda(\{1,2\})))$$
$$c_2 = (1 - (\lambda(\{1\}) + \lambda(\{1,2\}))).$$

Covariance of the joint distribution with zero mean

The covariance of zero mean, otherwise expressed as the covariance plus the product of the means is defined as

$$E(XY) = \sum_{j=0}^{\infty} \sum_{i=0}^{\infty} ij\pi_{i,j}.$$

The joint distribution $\pi_{i,j}$ is split into three parts, we therefore split the summation into three corresponding parts.

$$E(XY) = \sum_{j=0}^{\infty} \left(\sum_{i=0}^{j-1} ij\pi_{i,j} + j^2\pi_{j,j} + \sum_{i=j+1}^{\infty} ij\pi_{i,j} \right)$$

For clarity, we solve the summations part by part.

$$\begin{aligned}
\sum_{j=0}^{\infty}\sum_{i=0}^{j-1} ij\pi_{i,j} &= \sum_{y=0}^{\infty}\sum_{i=0}^{y-1} ij\frac{\lambda(\{1\})(1-c_1)}{c_1}\lambda(\emptyset)^i\frac{c_1^j}{c_1^i} \\
&= \frac{\lambda(\{1\})(1-c_1)}{c_1}\sum_{j=0}^{\infty} jc_1^j\sum_{i=0}^{j-1} i\left(\frac{\lambda(\emptyset)}{c_1}\right)^i \\
&= \frac{\lambda(\{1\})(1-c_1)}{c_1(\frac{\lambda(\emptyset)}{c_1}-1)^2}\sum_{j=0}^{\infty} jc_1^j\Bigg(j\left(\frac{\lambda(\emptyset)}{c_1}\right)^j(\frac{\lambda(\emptyset)}{c_1}-1) \\
&\qquad -\left(\frac{\lambda(\emptyset)}{c_1}\right)^j\frac{\lambda(\emptyset)}{c_1}+\frac{\lambda(\emptyset)}{c_1}\Bigg) \\
&= \frac{\lambda(\{1\})(1-c_1)}{c_1(\frac{\lambda(\emptyset)}{c_1}-1)^2}\Bigg(-\frac{\lambda(\emptyset)(\lambda(\emptyset)+1)(\frac{\lambda(\emptyset)}{c_1}-1)}{(\lambda(\emptyset)-1)^3} \\
&\qquad -\frac{\lambda(\emptyset)^2}{c_1(\lambda(\emptyset)-1)^2}+\frac{\lambda(\emptyset)}{(c_1-1)^2}\Bigg)
\end{aligned}$$

If we use

$$\sum_{j=0}^{\infty}\sum_{i=j+1}^{\infty} ij\pi_{i,j} = \sum_{i=0}^{\infty}\sum_{j=0}^{i-1} ij\pi_{i,j}$$

and the above result for the first part of the summation, we can analogously derive the expression for the last part of the summation, where $i > j$.

$$\begin{aligned}
\sum_{j=0}^{\infty}\sum_{i=j+1}^{\infty} ij\pi_{i,j} &= \frac{\lambda(\{2\})(1-c_2)}{c_2(\frac{\lambda(\emptyset)}{c_2}-1)^2}\Bigg(-\frac{\lambda(\emptyset)(\lambda(\emptyset)+1)(\frac{\lambda(\emptyset)}{c_2}-1)}{(\lambda(\emptyset)-1)^3} \\
&\qquad -\frac{\lambda(\emptyset)^2}{c_2(\lambda(\emptyset)-1)^2}+\frac{\lambda(\emptyset)}{(c_2-1)^2}\Bigg)
\end{aligned}$$

The simpler term of the summation is the second term, where $i = j$, which can be expressed as

$$\sum_{j=0}^{\infty} j^2\pi_{j,j} = \sum_{j=0}^{\infty}\lambda(\{1,2\})j^2\lambda(\emptyset)^j = \lambda(\{1,2\})\frac{-\lambda(\emptyset)(\lambda(\emptyset)+1)}{(\lambda(\emptyset)-1)^3}.$$

The summation of these three terms is equal to $E(XY)$.

$$\begin{aligned}E(XY) = \frac{\lambda(\{1\})(1-c_1)}{c_1(\frac{\lambda(\emptyset)}{c_1}-1)^2}\Bigg(-\frac{\lambda(\emptyset)(\lambda(\emptyset)+1)(\frac{\lambda(\emptyset)}{c_1}-1)}{(\lambda(\emptyset)-1)^3} - \frac{\lambda(\emptyset)^2}{c_1(\lambda(\emptyset)-1)^2} \\ + \frac{\lambda(\emptyset)}{(c_1-1)^2}\Bigg) + \lambda(\{1,2\})\frac{-\lambda(\emptyset)(\lambda(\emptyset)+1)}{(\lambda(\emptyset)-1)^3} \\ + \frac{\lambda(\{2\})(1-c_2)}{c_2(\frac{\lambda(\emptyset)}{c_2}-1)^2}\Bigg(-\frac{\lambda(\emptyset)(\lambda(\emptyset)+1)(\frac{\lambda(\emptyset)}{c_2}-1)}{(\lambda(\emptyset)-1)^3} - \frac{\lambda(\emptyset)^2}{c_2(\lambda(\emptyset)-1)^2} \\ + \frac{\lambda(\emptyset)}{(c_2-1)^2}\Bigg)\end{aligned}$$

Expected value of the marginal distributions

Recall that the marginal distributions were given as

$$\begin{aligned}\pi_i^{(X)} &= (1-(\lambda(\{1\})+\lambda(\{1,2\})))^i(\lambda(\{1\})+\lambda(\{1,2\})) = (1-c_2)c_2^i \\ \pi_j^{(Y)} &= (1-(\lambda(\{2\})+\lambda(\{1,2\})))^j(\lambda(\{2\})+\lambda(\{1,2\})) = (1-c_1)c_1^j.\end{aligned}$$

The expected value of the two discrete marginal distributions are obtained as follows

$$\begin{aligned}E(X) &= \sum_{i=0}^{\infty} i\pi_i^{(X)} = (1-c_2)\sum_{i=0}^{\infty} ic_2^i = \frac{c_2}{1-c_2} \\ E(Y) &= \sum_{j=0}^{\infty} j\pi_j^{(Y)} = (1-c_1)\sum_{j=0}^{\infty} jc_1^j = \frac{c_1}{1-c_1}.\end{aligned}$$

Covariance

A measure of interference of two random variables X and Y is the covariance. Covariance is defined as

$$\begin{aligned}\mathrm{Cov}(X,Y) &= E((X-E(X))(Y-E(Y))) \\ &= E(XY - XE(Y) - YE(X) + E(X)E(Y)) \\ &= E(XY) - E(X)E(Y).\end{aligned}$$

If we now use the expected value of the joint distribution and the marginal distribution, we get

$$\begin{aligned}\mathrm{Cov}(X,Y) = \frac{\lambda(\{1\})(1-c_1)}{c_1(\frac{\lambda(\emptyset)}{c_1}-1)^2}\Bigg(-\frac{\lambda(\emptyset)(\lambda(\emptyset)+1)(\frac{\lambda(\emptyset)}{c_1}-1)}{(\lambda(\emptyset)-1)^3} - \frac{\lambda(\emptyset)^2}{c_1(\lambda(\emptyset)-1)^2} \\ + \frac{\lambda(\emptyset)}{(c_1-1)^2}\Bigg) + \lambda(\{1,2\})\frac{-\lambda(\emptyset)(\lambda(\emptyset)+1)}{(\lambda(\emptyset)-1)^3} \\ + \frac{\lambda(\{2\})(1-c_2)}{c_2(\frac{\lambda(\emptyset)}{c_2}-1)^2}\Bigg(-\frac{\lambda(\emptyset)(\lambda(\emptyset)+1)(\frac{\lambda(\emptyset)}{c_2}-1)}{(\lambda(\emptyset)-1)^3} - \frac{\lambda(\emptyset)^2}{c_2(\lambda(\emptyset)-1)^2} \\ + \frac{\lambda(\emptyset)}{(c_2-1)^2}\Bigg) - \frac{c_2}{1-c_2}\frac{c_1}{1-c_1}.\end{aligned}$$

Using the symbolic toolbox in Matlab, we can simplify this expression to

$$\mathrm{Cov}(X,Y) = \frac{\lambda(\emptyset)\lambda(\{1,2\}) - \lambda(\{1\})\lambda(\{2\})}{(\lambda(\{1\})+\lambda(\{1,2\}))(\lambda(\{2\})+\lambda(\{1,2\}))(1-\lambda(\emptyset))}$$

Appendix B

M-file: Function for computing $\lambda_j(B)$

This M-file computes the probability of successful reception for networks that have at most one coupled node. Note that some lines were broken off, such that they fit on the page.

```
1  function [ lambdaSet lambdaProb ] = lambdaLinks(link,coupled,
2                                                   nodeTransProb,K)
3  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4  % Computes lambda for links, the input arguments are
5  % o link = [ ID From To Prob ], the last two should always be dummy
6  %       links with p = 1 and their correct q.
7  % o coupled are sets of nodes (supplied as a cell) that are connected to
8  %       the same transmitting node.
9  % o nodeTransProb is the probability 'q' of transmission for all nodes,
10 %       note that we include the source, node 0, as the first element of
11 %       this vector and therefore all indices are shifted by 1.
12 % o K is a cell with neighbouring nodes for all nodes
13 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
14
15 Nlink = link(:,1);
16 linkFrom = link(:,2); linkTo = link(:,3); linkProb = link(:,4);
17
18 lambdaSet = nchoose(Nlink);
19 lambdaProb = zeros(numel(lambdaSet),1);
20
21 for i = 1:numel(lambdaSet)
22
23     B = lambdaSet{i};
24
25     clear DSet DSubSet
26
27     DSubSet = nchoose(setxor(Nlink,B)); DSubSet(end+1) = {[]}; %#ok<AGROW>
28
```

```
29      DSet = cell(numel(DSubSet),1);
30      for ii = 1:numel(DSubSet)
31          DSet(ii,:) = {[ B DSubSet{ii} ]};
32      end
33
34      for iii = 1:numel(coupled)
35          DSet = cellfun(@(c)addTx(c,coupled{iii}),DSet,
36                                              'UniformOutput',false);
37      end
38
39      DSet = uniqueCellArray(DSet);
40
41      for j = 1:numel(DSet)
42          D = DSet{j};
43
44          setTx = unique(linkFrom(D));
45          setNoTx = setxor(linkFrom,setTx);
46          setRx = B;
47          tmpSet = [];
48          for jj = 1:numel(setTx)
49              tmpSet = [ tmpSet find(linkFrom == setTx(jj))' ]; %#ok<AGROW>
50          end
51          setNoRx = setxor(setRx,tmpSet);
52
53          tmp = 1;
54          for k = 1:length(setNoTx)
55              tmp = tmp*(1-nodeTransProb(setNoTx(k)+1));
56          end
57          for k = 1:length(setTx)
58              tmp = tmp*nodeTransProb(setTx(k)+1);
59          end
60          for k = 1:length(setRx)
61              tmp = tmp*rInt(setRx(k));
62          end
63          for k = 1:length(setNoRx)
64              tmp = tmp*(1-rInt(setNoRx(k)));
65          end
66
67          lambdaProb(i) = lambdaProb(i) + tmp;
68
69      end
70
71  end
72
73  lambdaProb(end+1) = 1 - sum(lambdaProb);
74  lambdaSet(end+1) = {[]};
75
76  [ lambdaSet lambdaProb ] = lambdaSmallerNetwork(lambdaSet,
77                                      lambdaProb,Nlink(1:(end-1)),Nlink);
78
79      function pInt = rInt(j)
80
81          if ismember(linkTo(j),linkFrom(D))
82              pInt = 0;
```

```
83          else
84              L = intFact;
85              pInt = L*linkProb(j);
86          end
87
88          function L = intFact
89              L = 1/(numel(intersect(D,K{linkTo(j)}))+1);
90          end
91
92      end
93
94  end
```

Appendix C

M-file: Computing the 4-dimensional iterate

This M-file computes the main 4-dimensional iterate. Note that some lines were broken off, such that they fit on the page.

```
function piJoint = jointDist4DArbitraryFast(marg2DBothBranches,
                        marg2DBranch1,marg2DBranch2,marg3DBranch1,
                        marg3DBranch2,marg4D,
                        g0,g1,g2,g3,g4,g12,g14,g23,g34,M)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function computes the joint distribution of 4 nodes in a system with
% 2 branches. There are 2 nodes on each branch and the nodes on a branch
% have to be succeeding nodes.
%
% The nodes are labeled as follows
% o Node 1 is the node closest to the source on the first branch;
% o Node 2 is the node closest to the source on the second branch;
% o Node 3 is the succeeding node of node 1;
% o Node 4 is the succeeding node of node 2;
%
% Note that this only gives us an approximation of the joint distribution,
% by increasing the size of the bounding box (M), the approximation becomes
% more accurate. A measure for the accuracy of the approximation is the
% summation of the 4-dimensional joint distribution, the closer to 1 this
% is, the more accurate it is.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% We need the conditional marginal distributions to compute the transition
% probabilities.

% Conditional distribution of ( node before 1 | node 1 )
cond2DBranch1 = zeros(M,M);
for j = 1:M
    if sum(marg2DBranch1(:,j)) ~= 0
        cond2DBranch1(:,j) = marg2DBranch1(:,j)./sum(marg2DBranch1(:,j));
```

```
    end
end

% Conditional distribution of ( node before 2 | node 2 )
cond2DBranch2 = zeros(M,M);
for j = 1:M
    if sum(marg2DBranch2(:,j)) ≠ 0
        cond2DBranch2(:,j) = marg2DBranch2(:,j)./sum(marg2DBranch2(:,j));
    end
end

% Conditional distribution of ( node before 1, node before 2
%                                              | node 1, node 2 )
condMarg4D = zeros(M,M,M,M);
for k = 1:M
    for l = 1:M
        if sum(sum(marg4D(:,:,k,l))) ≠ 0
            condMarg4D(:,:,k,l) = marg4D(:,:,k,l)./sum(sum(marg4D(:,:,k,l)));
        end
    end
end

% Preallocating memory space for the b vector.
b = zeros(M^4,1);

% Preallocating memory for the vectors that will later construct the sparse
% matrix A. The triplet (iA,jA,sA) gives us a row, a column and a value,
% respectively.
% Constructing an empty sparse A matrix and indexing into this matrix is
% also an option, but that appeared to be very slow.
iA = zeros(round(0.8*M^5),1); jA = zeros(round(0.8*M^5),1);
sA = zeros(round(0.8*M^5),1);

% Current index that we are using to index into the vectors iA, jA and sA.
indSparseVects = 0;

% Vector used as an alternative to the find() function to speed up the
% program; find() is slow for large vectors.
indJrow = 1:M^4;

% Here we start the construction of the matrix A by adding elements to the
% vectors iA, jA and sA.
for l = 1:(M-1)
    tic
    for j = 1:(l+1)
        for k = 1:(M-1)
            for i = 1:(k+1)

                % The variable 'id' states in what row we are going to
                % store the obtained equilibrium equation in the matrix A.
                % This in done in a way, such that it corresponds to the
                % lexicographical structure of the vector x.
                id = 1 + (j - 1) + (i - 1)*M + (k - 1)*M^2 + (l - 1)*M^3;

```

```
85                % The state space is split into 25 different regions, each
86                % region has its own incoming transitions. We call a
87                % nested subfunction tr4D (transition probabilities in
88                % 4 dimensions) to compute a J matrix and an entry in the b
89                % vector.
90                % The J matrix is M-by-M-by-M-by-M, it contains the
91                % coefficients of the equilibrium equation in matrix form.
92                if isequal([ i j k l ],[ 2 2 1 1 ]) % 7
93                    [ J b(id) ] = tr4D([ 0 0 0 1 0 0 0 0 0 ]);
94                elseif isequal([ i j k ],[ 2 1 1 ]) && l ≥ 2 % 8
95                    [ J b(id) ] = tr4D([ 0 0 1 0 0 0 0 0 0 ]);
96                elseif isequal([ i k ],[ 2 1 ]) && j == (l+1) && j ≥
    3 % 9
97                    [ J b(id) ] = tr4D([ 0 0 1 1 0 0 1 0 0 ]);
98                elseif isequal([ i k ],[ 2 1 ]) && l ≥ j && j ≥ 2 % 10
99                    [ J b(id) ] = tr4D([ 0 0 1 0 0 0 1 0 0 ]);
100               elseif isequal([ i j l ],[ 1 2 1 ]) && k ≥ 2 % 12
101                   [ J b(id) ] = tr4D([ 0 1 0 0 0 0 0 0 0 ]);
102               elseif isequal([ i j ],[ 1 1 ]) && k ≥ 2 && l ≥ 2 % 13
103                   [ J b(id) ] = tr4D([ 1 0 0 0 0 0 0 0 0 ]);
104               elseif i == 1 && k ≥ 2 && j == (l+1) && j ≥ 3 % 14
105                   [ J b(id) ] = tr4D([ 1 1 0 0 1 0 0 0 0 ]);
106               elseif i == 1 && k ≥ 2 && l ≥ j && j ≥ 2 % 15
107                   [ J b(id) ] = tr4D([ 1 0 0 0 1 0 0 0 0 ]);
108               elseif i == (k+1) && i ≥ 3 && isequal([ j l ],[ 2 1 ]) % 17
109                   [ J b(id) ] = tr4D([ 0 1 0 1 0 0 0 1 0 ]);
110               elseif i == (k+1) && i ≥ 3 && j == 1 && l ≥ 2 % 18
111                   [ J b(id) ] = tr4D([ 1 0 1 0 0 1 0 0 0 ]);
112               elseif i == (k+1) && i ≥ 3 && j == (l+1) && j ≥ 3 % 19
113                   [ J b(id) ] = tr4D([ 1 1 1 1 1 1 1 1 1 ]);
114               elseif i == (k+1) && i ≥ 3 && l ≥ j && j ≥ 2 % 20
115                   [ J b(id) ] = tr4D([ 1 0 1 0 1 1 1 0 1 ]);
116               elseif k ≥ i && i ≥ 2 && isequal([ j l ],[ 2 1 ]) % 22
117                   [ J b(id) ] = tr4D([ 0 1 0 0 0 0 0 1 0 ]);
118               elseif k ≥ i && i ≥ 2 && j == 1 && l ≥ 2 % 23
119                   [ J b(id) ] = tr4D([ 1 0 0 0 0 1 0 0 0 ]);
120               elseif k ≥ i && i ≥ 2 && j == (l+1) && j ≥ 3 % 24
121                   [ J b(id) ] = tr4D([ 1 1 0 0 1 1 0 1 1 ]);
122               elseif k ≥ i && i ≥ 2 && l ≥ j && j ≥ 2 % 25
123                   [ J b(id) ] = tr4D([ 1 0 0 0 1 1 0 0 1 ]);
124               else
125                   [ J b(id) ] = tr4D([ 0 0 0 0 0 0 0 0 0 ]);
126               end
127
128               % The matrix J has to rearranged into a row and the
129               % non-zero elements (J only contains non-negative values)
130               % are stored in sA, the location of the element is stored
131               % in iA (row) and jA (column).
132               Jrow = J(:);
133               JrowPos = indJrow(Jrow > 0);
134               JrowPosNum = numel(JrowPos);
135
136               iA( (indSparseVects+1):(indSparseVects+JrowPosNum) ) =
137                                       id*ones(1,JrowPosNum);
```

```
138                 jA( (indSparseVects+1):(indSparseVects+JrowPosNum) ) =
139                                         JrowPos;
140                 sA( (indSparseVects+1):(indSparseVects+JrowPosNum) ) =
141                                         Jrow(JrowPos);
142
143                 % We update the index used to store the positive values of
144                 % J in iA, jA and sA.
145                 indSparseVects = indSparseVects + JrowPosNum;
146
147             end
148         end
149     end
150
151     % Iteration progress and memory usage is printed to the command window
152     display(sprintf('Iteration %d out of %d \t--\t %0.5g seconds\n'
153                 'Storage used \t %0.5g, \t out of \t %0.5g\n'
154                 ,l,M-1,toc,nnz(iA),length(iA)))
155
156 end
157
158 % Clear all temporary files, such that we have more memory available for
159 % the matrix A to be added to the memory.
160 clear Jrow JrowPos JrowPosNum indSparseVects J
161 clear condMarg4D cond2DBranch1 cond2DBranch2
162
163 % Construct the matrix A by using the indices iA and jA and the
164 % corresponding values sA. We subtract the value 1 from each row, as we
165 % want all variables x on one side.
166 % (The system is actually Ax - b = x, which leads to (A-I)x = b )
167 A = sparse(iA(1:nnz(iA)),jA(1:nnz(jA)),sA(1:nnz(sA)),M^4,M^4);
168 A = A - spdiags(ones(M^4,1),0,M^4,M^4);
169
170 % Solve the system
171 x = A\b;
172
173 % Reconstruct the 4-dimensional joint distribution by obtaining the i, j, k
174 % and l value from the row number.
175 piJoint = zeros(M,M,M,M);
176 for n = 1:M^4
177     l = ceil(n/M^3);
178     k = ceil( (n - (l-1)*M^3) / M^2 );
179     i = ceil( (n - (l-1)*M^3 - (k-1)*M^2 ) / M );
180     j = n - (l-1)*M^3 - (k-1)*M^2 - (i-1)*M;
181     piJoint(i,j,k,l) = x(n);
182 end
183
184 %% Nested function to compute the transition probabilities
185
186     function [ J, b ] = tr4D(r)
187     % The variable r is a list describing type of incoming transitions
188     % [ lambda(1,2) lambda(1,4) lambda(2,3) lambda(3,4) ...
189     % lambda(1) lambda(2) lambda(3) lambda(4) lambda(\emptyset) ]
190
191     % Preallocating memory for J and b.
```

```
192     J = zeros(M,M,M,M);
193     b = 0;
194 
195     % Indices used for the conditional distribution, this has to be used
196     % due to the structure of the equilibrium equations.
197     iB = max(i-1,1):k; jB = max(j-1,1):l;
198 
199     % We check for all possible incoming transitions and compute the ones
200     % that are needed for that region.
201     if r(1)
202         J(jB,iB,k-1,l-1) = J(jB,iB,k-1,l-1) + ...
203             g12*reshape(condMarg4D(i,j,iB,jB),numel(iB),numel(jB))';
204     end
205     if r(2)
206         b = b - g14 * cond2DBranch1(i,iB) * marg3DBranch1(iB,j-1,k-1);
207     end
208     if r(3)
209         b = b - g23 * cond2DBranch2(j,jB) * marg3DBranch2(i-1,jB,l-1)';
210     end
211     if r(4)
212         b = b - g34 * marg2DBothBranches(i-1,j-1);
213     end
214     if r(5)
215         J(j-1,iB,k-1,l-1) = J(j-1,iB,k-1,l-1) + g1 * cond2DBranch1(i,iB);
216     end
217     if r(6)
218         J(jB,i-1,k-1,l-1) = J(jB,i-1,k-1,l-1) + g2 * cond2DBranch2(j,jB)';
219     end
220     if r(7)
221         b = b - g3 * marg3DBranch2(i-1,j-1,l-1);
222     end
223     if r(8)
224         b = b - g4 * marg3DBranch1(i-1,j-1,k-1);
225     end
226     if r(9)
227         J(j-1,i-1,k-1,l-1) = J(j-1,i-1,k-1,l-1) + g0;
228     end
229 
230     end
231 
232 end
```

Appendix D

M-file: Using the iteration scheme

This M-file is used to run the iteration scheme, it calls all the necessary functions. In the current state, it runs the example shown in the proof of concept section. Note that some lines were broken off, such that they fit on the page.

```
%% Iteration run file
%
% We look at a model consisting of 2 branches, both connected to the same
% source.
% o The node closest to the source on the first branch is node 1;
% o The node closest to the source on the second branch is node 2;
% o The succeeding node on the first branch is node 3;
% o The succeeding node on the second branch is node 4.
%
% This file iteratively computes a 4-dimensional joint distribution of two
% succeeding nodes on the first branch and two succeeding nodes on the
% second branch.

clc, clear all

%% System parameters

% Homogenous system
p = 0.9;
q = 0.5;

% Specifying the links [ ID from to prob ]
link = [  1  0 1  p ;
          2  0 2  p ;
          3  1 3  p ;
          4  2 4  p ;
          5  3 5  p ;
```

```
28            6  4 6  p ;
29            7  5 7  p ;
30            8  6 8  p ;
31            9  7 9  p ;
32            10 8 9  p ;
33            11 9 10 1 ];  % dummy link
34
35  Nmax = link(end,1);  N = 1:(Nmax - 1);
36
37  Nhops = 4;
38
39  % The two links that are coupled at the source
40  coupled = { [ 1 2 ] };
41
42  % Probability of transmission for each node
43  nodeTransProb = ones(Nmax+1,1)*q;
44
45  % Neighbouring nodes that affect successful reception probabilities
46  K = cell(Nmax,1);
47
48  [ lambdaSet lambdaProb ] = lambdaLinks(link,coupled,nodeTransProb,K);
49
50  %% Iteration parameters
51
52  % Size of the bounding box
53  M = 20;
54
55  %% Creating the input for the iteration scheme
56
57  % Reception probabilities for the first four nodess of the system
58  [ lambdaSetCurr lambdaProbCurr ] = lambdaSmallerNetwork(lambdaSet,
59                                                  lambdaProb,[1 2 3 4],N);
60
61  l0 = lambdaProbCurr(fI(lambdaSetCurr,[]));
62  l1 = lambdaProbCurr(fI(lambdaSetCurr,[1]));
63  l2 = lambdaProbCurr(fI(lambdaSetCurr,[2]));
64  l3 = lambdaProbCurr(fI(lambdaSetCurr,[3]));
65  l4 = lambdaProbCurr(fI(lambdaSetCurr,[4]));
66  l12 = lambdaProbCurr(fI(lambdaSetCurr,[1 2]));
67  l14 = lambdaProbCurr(fI(lambdaSetCurr,[1 4]));
68  l23 = lambdaProbCurr(fI(lambdaSetCurr,[2 3]));
69  l34 = lambdaProbCurr(fI(lambdaSetCurr,[3 4]));
70
71  % Joint distribution of ( node 1, node 2)
72  marg2DBothBranches = jointDistSameTx(l0+l3+l4+l34,l1+l14,l2+l23,l12,M);
73
74  % Joint distribution of ( source, node 1 )
75  marg2DBranch1 = zeros(M,M);
76  marg2DBranch1(1,:) = sum(marg2DBothBranches,2);
77
78  % Joint distribution of ( source, node 2 )
79  marg2DBranch2 = zeros(M,M);
80  marg2DBranch2(1,:) = sum(marg2DBothBranches,2);
81
```

```
% Joint distribution of ( source, source, node 1, node 2 )
marg4D = zeros(M,M,M,M);
marg4D(1,1,:,:) = marg2DBothBranches;

% Joint distribution of ( node 1, node 2, node 3 )
marg3DBranch1 = jointDist3DArbitraryFast(marg2DBothBranches,
                    marg2DBranch1,marg2DBranch2,marg4D,
                    l0+l4,l1+l14,l2,l3+l34,l12,l23,M);

% Joint distribution of ( node 1, node 2, node 4 )
marg3DBranch2 = jointDist3DArbitraryFastAlt(marg2DBothBranches,
                    marg2DBranch1,marg2DBranch2,marg4D,
                    l0+l3,l1,l2+l23,l4+l34,l12,l14,M);

%% First iterate

display(sprintf('\n\t Iterate number: \t %g',1))

% Joint disitrubtion of ( node 1, node 2, node 3, node 4 )
marg4DIter = jointDist4DArbitraryFast(marg2DBothBranches,marg2DBranch1,
                        marg2DBranch2,marg3DBranch1,marg3DBranch2,marg4D,
                        l0,l1,l2,l3,l4,l12,l14,l23,l34,M);

sum(sum(sum(sum(marg4DIter))))

%% Obtaining the input for the next iterate

for iter = 2:Nhops

    % Number of the links we are considering for this iterate
    n = [ 2*(iter-1)+1 2*(iter-1)+2 2*(iter-1)+3 2*(iter-1)+4 ];

    % Reception probabilities for the current iterate
    [ lambdaSetCurr lambdaProbCurr ] =
                    lambdaSmallerNetwork(lambdaSet,lambdaProb,n,N);

    l0 = lambdaProbCurr(fI(lambdaSetCurr,[]));
    l1 = lambdaProbCurr(fI(lambdaSetCurr,[ n(1) ]));
    l2 = lambdaProbCurr(fI(lambdaSetCurr,[ n(2) ]));
    l3 = lambdaProbCurr(fI(lambdaSetCurr,[ n(3) ]));
    l4 = lambdaProbCurr(fI(lambdaSetCurr,[ n(4) ]));
    l12 = lambdaProbCurr(fI(lambdaSetCurr,[ n(1) n(2) ]));
    l14 = lambdaProbCurr(fI(lambdaSetCurr,[ n(1) n(4) ]));
    l23 = lambdaProbCurr(fI(lambdaSetCurr,[ n(2) n(3) ]));
    l34 = lambdaProbCurr(fI(lambdaSetCurr,[ n(3) n(4) ]));

    % Setting up for a new iterate
    marg4DIterPrev = marg4DIter;

    % 2-dimensional joint distributions
    marg2DBothBranches = squeeze(sum(sum(marg4DIterPrev,2),1));
    marg2DBranch1 = squeeze(sum(sum(marg4DIterPrev,4),2));
    marg2DBranch2 = squeeze(sum(sum(marg4DIterPrev,3),1));

```

```
136     % 3-dimensional joint distributions
137     marg3DBranch1 = jointDist3DArbitraryFast(marg2DBothBranches,
138                                 marg2DBranch1,marg2DBranch2,marg4D,
139                                 l0+l4,l1+l14,l2,l3+l34,l12,l23,M);
140     marg3DBranch2 = jointDist3DArbitraryFastAlt(marg2DBothBranches,
141                                 marg2DBranch1,marg2DBranch2,marg4D,
142                                 l0+l3,l1,l2+l23,l4+l34,l12,l14,M);
143
144     % New iterate
145     display(sprintf('\n\t Iterate number: \t %g',iter))
146     marg4DIter = jointDist4DArbitraryFast(marg2DBothBranches,
147                     marg2DBranch1,marg2DBranch2,marg3DBranch1,
148                     marg3DBranch2,marg4DIterPrev,
149                     l0,l1,l2,l3,l4,l12,l14,l23,l34,M);
150
151     sum(sum(sum(sum(marg4DIter))))
152 end
```