Ant Algorithms and Generalized Finite Urns

by

Christopher Leith

A thesis submitted to the
Department of Mathematics and Statistics
in conformity with the requirements for
the degree of Doctor of Philosophy

Queen's University
Kingston, Ontario, Canada
July 2005

Abstract

Researchers in various fields are constructing algorithms that mimic the desirable collective behaviour of pheromone trail laying biological agents. These ant algorithms, as they are commonly called, can be used to solve complex optimization problems, or act as a distributed control layer in a dynamic system. Our interest lies mainly with the latter application.

Analyzing the effectiveness of ant-based algorithms is typically conducted by computer simulation alone. There are few mathematical studies that attempt to capture their seemingly random yet controlled behaviour. In this work, we build probabilistic models that guide our exploration of ant-based algorithms. We develop a generalized finite urn and investigate it in detail. We then abstract this base model into a Markovian process that is better suited to studying a dynamic environment, such as ant-based routing. Later, harnessing our mathematical findings, we propose a new framework for a general class of ant algorithms. Specifically, we introduce additional data structures that allow us increased control over the decoupling of ant actions and the underlying system that they indirectly manage. Finally, to complement our mathematical approach, we also evaluate the performance of our modified ant algorithm implementation through computer simulation.

Acknowledgements

A project this large necessarily relies on a group of dedicated people. Some get involved by their own choice, while others come by association to the candidate. Each play a vital role.

I want to thank my supervisor, Dr. Glen Takahara, for his guidance on this academic journey. He has been a willing and inspiring mentor at all stages of this work.

My parents, Steve and Dianne Leith, also deserve enormous credit for having patience and respect for my endeavour, sometimes with little feedback to secure their continued interest. I could not have completed the work without their support.

Finally, I wish to thank Maya Mukhida. Although her involvement began part of the way through this undertaking, her encouragement toward the end was more important to me than any other.

Statement of Originality

I certify that the ideas presented in this document represent, to the best of my knowledge, original research. All related work is referenced according to standard practices of the discipline. Finally, I acknowledge the generous support of my supervisor, Dr. Glen Takahara.

Contents

List of Tables

List of Figures

List of Symbols

C = number of colour types.

M = finite urn memory size.

n = discrete time index.

t = continuous time index.

$\mathcal{G}$ = set of general form urns.

$\mathcal{C}$ = set of constant weight form urns.

$\mathcal{E}$ = set of exchangeable form urns.

$\mathcal{D}$ = set of definite form urns.

$\mathcal{L}$ = set of linear form urns.

$W_i(n, M)$ = weight of colour type i at time n in an urn with memory M and without a null ball.

$W(n, M) = (W_0(n, M), W_1(n, M), \ldots, W_{C-1}(n, M))$ = vector of $W_i(n, M)$ values; urn composition at time n in an urn with memory M and without a null ball.

$W'_i(n, M)$ = weight of colour type i at time n in an urn with memory M and with a null ball.

$W'(n, M) = (W'_0(n, M), W'_1(n, M), \ldots, W'_{C-1}(n, M))$ = vector of $W'_i(n, M)$ values; urn composition at time n in an urn with memory M and with a null ball.

$X_m(n, M)$ = colour type in slot m at time n in an urn with memory M and without a null ball.

$X(n, M) = (X_0(n, M), X_1(n, M), \ldots, X_{M-1}(n, M))$ = vector of $X_m(n, M)$ values; internal feedback at time n in an urn with memory M and without a null ball; discrete time functional weight finite urn without a null ball.

$X'_m(n, M)$ = colour type in slot m at time n in an urn with memory M and with a null ball.

$X'(n, M) = (X'_0(n, M), X'_1(n, M), \ldots, X'_{M-1}(n, M))$ = vector of $X'_m(n, M)$ values; internal feedback at time n in an urn with memory M and with a null ball; discrete time functional weight finite urn with a null ball.

$S_i(n, M)$ = number of slots of colour type i at time n in an urn with memory M and without a null ball.

$S(n, M) = (S_0(n, M), S_1(n, M), \ldots, S_{C-1}(n, M))$ = vector of $S_i(n, M)$ values.

$S'_i(n, M)$ = number of slots of colour type i at time n in an urn with memory M and with a null ball.

$S'(n, M) = (S'_0(n, M), S'_1(n, M), \ldots, S'_{C-1}(n, M))$ = vector of $S'_i(n, M)$ values.

α_i = base weight of colour type i.

$\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_{C-1})$ = vector of α_i values.

Δ_{ij} = weight parameter of colour type i given colour type j selected.

$\Delta = (\Delta_{ij})_{ij}$ = weight matrix.

$\Delta_{RS,i} = \sum_{j=0}^{C-1} \Delta_{ij}$ = sum of the i^{th} row of Δ.

$\Delta_{CS,j} = \sum_{i=0}^{C-1} \Delta_{ij}$ = sum of the j^{th} column of Δ.

$\Delta_{CS} = (\Delta_{CS,0}, \Delta_{CS,1}, \ldots, \Delta_{CS,C-1})$ = vector of $\Delta_{CS,j}$ values.

$\Delta_{MCS} = \max_j\{\Delta_{CS,j}\}$ = maximum column sum of Δ.

Δ_{CCS} = constant column sum of Δ in a constant weight form urn.

δ_i = off-diagonal entry in row i in a definite form urn.

$D = (\delta_i)_{ij}$ = repeated column matrix of off-diagonal entries in a definite form urn.

$\tau_i = (\Delta_{ii} - \delta_i)$ = on-diagonal, off-diagonal difference in a definite form urn.

$T = (\tau_i)_{ii}$ = diagonal matrix of on-diagonal, off-diagonal differences in a definite form urn.

x_m = colour type selected in the m^{th} most recent draw in an urn.

x = vector of x_m values; internal feedback portion of state in an urn.

$w_i(x)$ = weight of colour type i corresponding to x in an urn.

$w(x)$ = total weight corresponding to x in an urn without a null ball.

w' = total weight corresponding to x in an urn with a null ball.

s_i = number of slots of colour type i in an urn.

$s = (s_0, s_1, \ldots, s_{C-1})$ = vector of s_i values.

$g_i(x)$ = general functional weight of colour type i corresponding to x in an urn.

$\tilde{x}_{(i)}$ = next vector of internal feedback following x and assuming colour type i selected.

$\tilde{x}^{(i)}$ = previous vector of internal feedback preceding x and assuming colour type i is dropped.

$p_{x,\tilde{x}}(M)$ = stationary probability of a transition from x to $\tilde{x}$ in an urn with memory M without a null ball.

$P(M)$ = transition probability matrix of an urn with memory M without a null ball.

$p'_{x,\tilde{x}}(M)$ = stationary probability of a transition from x to $\tilde{x}$ in an urn with memory M with a null ball.

$P'(M)$ = transition probability matrix of an urn with memory M with a null ball.

$\pi_{X,x}(M)$ = stationary probability of state x in a functional weight finite urn with memory M without a null ball.

$\pi_{X',x}(M)$ = stationary probability of state x in a functional weight finite urn with memory M with a null ball.

$\pi_{S,s}(M)$ = stationary probability of state s in a functional weight finite urn with memory M without a null ball.

$\pi_{S',s}(M)$ = stationary probability of state s in a functional weight finite urn with memory M with a null ball.

$G(M)^{-1}$ = normalization constant of stationary distribution in a functional weight finite urn with memory M without a null ball.

$G'(M)^{-1}$ = normalization constant of stationary distribution in a functional weight finite urn with memory M with a null ball.

$p_i(M)$ = probability of drawing colour type i in steady state in an urn with memory M without a null ball.

$p'_i(M)$ = probability of drawing colour type i in steady state in an urn with memory M with a null ball.

$q_i(M)$ = probability slot 1 has colour type i in steady state in an urn with memory M without a null ball.

$q'_i(M)$ = probability slot 1 has colour type i in steady state in an urn with memory M with a null ball.

$p_{i,j}(M)$ = probability of drawing colour type i, with colour type j currently in slot 1, in steady state in an urn with memory M without a null ball.

$p'_{i,j}(M)$ = probability of drawing colour type i, with colour type j currently in slot 1, in steady state in an urn with memory M with a null ball.

$q_{i,j}(M)$ = probability slot 1 has colour type i and slot 2 has colour type j in steady state in an urn with memory M without a null ball.

$q'_{i,j}(M)$ = probability slot 1 has colour type i and slot 2 has colour type j in steady state in an urn with memory M with a null ball.

$p'_{null}(M)$ = probability of drawing a null ball in steady state in an urn with memory M with a null ball.

$p'_{null,i}(M)$ = probability of drawing a null ball in steady state and colour type i in slot 1 in an urn with memory M with a null ball.

$p'_{null|i}(M)$ = probability of drawing a null ball in steady state given colour type i in slot 1 in an urn with memory M with a null ball.

$p'_{null,i,j}(M)$ = probability of drawing a null ball in steady state and colour type i in slot 1 and colour type j in slot 2 in an urn with memory M with a null ball.

$p'_{null|i,j}(M) = p'_{null,i,j}(M)/q'_{i,j}(M)$ = probability of drawing a null ball in steady state given colour type i in slot 1 and colour type j in slot 2 in an urn with memory M with a null ball.

$r'_i(M)$ = ratio of $(1 - p'_{null}(M))$ to $(1 - p'_{null|i}(M))$.

$r'_{i,j}(M)$ = ratio of $(1 - p'_{null}(M))$ to $(1 - p'_{null|ij}(M))$.

$\overline{W}_i(M)$ = centred and scaled weight of colour type i in an urn with memory M without a null ball.

$\overline{W}'_i(M)$ = centred and scaled weight of colour type i in an urn with memory M with a null ball.

$\overline{S}_i(M)$ = centred and scaled number of slots of colour type i in an urn with memory M without a null ball.

$\overline{S}'_i(M)$ = centred and scaled number of slots of colour type i in an urn with memory M with a null ball.

$\sigma_{i,j}(M)$ = scaled covariance of $S_i(M)$ and $S_j(M)$.

$\sigma'_{i,j}(M)$ = scaled covariance of $S'_i(M)$ and $S'_j(M)$.

$F_{ij}(M)$ = bivariate cumulative distribution function of $\overline{S}_i(M)$ and $\overline{S}_j(M)$.

$F'_{ij}(M)$ = bivariate cumulative distribution function of $\overline{S}'_i(M)$ and $\overline{S}'_j(M)$.

λ = largest real eigenvalue of Δ.

q = limiting scaled mean vector in a linear form urn with memory M without a null ball.

q' = limiting scaled mean vector in a linear form urn with memory M with a null ball.

Σ = matrix of limiting scaled covariances in a linear form urn without a null ball.

Σ' = matrix of limiting scaled covariances in a linear form urn with a null ball.

Σ_{IID} = matrix of limiting scaled covariances in an IID reference system.

I = identity matrix.

$L = \frac{1}{\lambda}(\Delta - q{\Delta_{CS}}^t)$ = auxiliary matrix for linear form solution of Σ.

R = arbitrary matrix with repeated columns.

$U_1 = \left(\frac{\lambda}{\lambda - \tau_i}\right)_{ij}$ = first auxiliary diagonal matrix for linear definite form solution of Σ.

$U_2 = \left(\frac{q_i}{\lambda - \tau_i}\Delta_{CS,j}\right)_{ij}$ = second auxiliary matrix for linear definite form solution of Σ.

μ = exponential rate of internal transitions in an embedded functional weight finite urn.

ν = exponential rate of external transitions in an embedded functional weight finite urn.

F = number of external feedback states of an embedded functional weight finite urn.

E = number of external process states of an embedded functional weight finite urn.

$Y_m(n, M)$ = external feedback in slot m at time n in an urn with memory M.

$Y(n, M) = (Y_0(n, M), Y_1(n, M), \ldots, Y_{M-1}(n, M))$ = vector of $Y_m(n, M)$ values; external feedback at time n in an urn with memory M.

$Z(n, M)$ = external process state at time n in an urn with memory M.

$(X(n, M), Y(n, M), Z(n, M))$ = discrete time embedded functional weight finite urn.

y_m = external feedback associated with the m^{th} most recent draw in an urn.

y = vector of y_m values; external feedback portion of a state in an urn.

z = external process state in an urn.

$w_i(x, y)$ = weight of type i corresponding to x and y in an urn.

$w(x, y)$ = total weight corresponding to x and y in an urn.

$g_i(x, y)$ = general functional weight of colour type i corresponding to x and y in an urn.

$f_(i, z)$ = external feedback assuming colour type i is selected and external process is in state z in an urn.

$\tilde{y}_{(i,z)}$ = next vector of external feedback following y and assuming colour type i selected and external process is in state z.

$\tilde{z}$ = next external state following z.

$\psi_{z,\tilde{z}}(x)$ = stationary probability of a transition from z to $\tilde{z}$, depending on x, in an urn.

$\Psi(x, M)$ = matrix of $\psi_{z,\tilde{z}}(x)$ values; transition probability matrix of external process component of an urn.

$p_{(x,y,z),(\tilde{x},\tilde{y},\tilde{z})}(M)$ = stationary probability of a transition from (x, y, z) to $(\tilde{x}, \tilde{y}, \tilde{z})$ in an urn with memory M.

$\pi_{(X,Y,Z),(x,y,z)}(M)$ = stationary probability of state (x, y, z) in an embedded functional weight finite urn with memory M.

$\mathcal{F}_i$ = set of external feedback states favourable to colour type i in an urn.

ρ = expected frequency of correct choices in an urn.

x_i^* = particular internal feedback state where each component is i.

y_i^* = particular external feedback state where each component is 1.

z_j^* = particular external state favouring the selection of colour type j.

$P^*(M)$ = special case transition probability matrix for transient measures.

$P_n^*(M)$ = n^{th} step special case transition probability matrix for transient measures.

$\mathcal{H}_j$ = set of states favouring the selection of colour type j in an urn.

$p_{i\to\mathcal{H}_j}(t)$ = probability of being in the set $\mathcal{H}_j$, having started in (x_i^*, y_i^*, z_j^*), at time t.

$p_{i\to j}(t)$ = probability of selecting colour type j, having started in (x_i^*, y_i^*, z_j^*), at time t.

V = number of nodes in the network.

K = number of outgoing links at a node.

$C(i)$ = number of routes to destination node i.

M_0 = maximum number of database entries used for next-hop routing table update.

M_1 = maximum number of database entries used for route routing table update.

(r_{im}, d_{im}, t_{im}) = m^{th} most recent piece of information in the database about trips to destination node i; r_{im} is the route; d_{im} is the delay estimate; t_{im} is the time-stamp.

$\mathcal{R}(i,k)$ = set of routes to destination i with first hop k.

$j_{0,ik}(m)$ = column index of next most recent piece of information in the database from column m about trips to destination node i using routes with first hop k.

$j_{1,ir}(m)$ = column index of next most recent piece of information in the database from column m about trips to destination node i using route r.

$P_0(t)$ = next-hop routing pheromone table used by ants.

$P_1(t)$ = route routing pheromone table used by data packets.

$p_{0,ik}(t)$ = pheromone level in next-hop routing table associated with destination node i and first hop k at time t.

$p_{1,ir}(t)$ = pheromone level in route routing table associated with destination node i and route r at time t.

$u_{0,ik}(m)$ = factor downplaying pheromone update to next-hop routing table entry corresponding to destination node i and first hop k based on time.

$u_{1,ir}(m)$ = factor downplaying pheromone update to route routing table entry corresponding to destination node i and route r based on time.

$v_{1,ir}(m)$ = factor downplaying pheromone update to route routing table entry corresponding to destination node i and route r based on delay estimate discrepancies.

Chapter 1

Introduction

One of the greatest human attributes is laziness. It ensures that we maximize our fortunes with the least amount of effort, by taking a moment to ponder the most efficient way to accomplish a given task. In some cases, this leads to complex optimization problems that we must solve before proceeding. In other instances, the system we wish to economize is changing so rapidly that the goal becomes that of just effectively controlling it over time. Either situation can present a major challenge without the proper tools.

To date various algorithmic solutions have been proposed for these types of problems. Care must be taken however, when selecting and evaluating the most appropriate algorithm. Beyond the basic measures of speed and correctness, we are often just as concerned with the algorithm's robustness, ability to adapt and potential to be scaled to larger problems. Few algorithms perform well in every category.

Another valuable human trait is our ability to learn by observation. In the next section we will see how researchers are learning to exploit the mechanisms underlying

natural systems to build a new class of algorithms with highly desirable properties [8].

1.1 Biological Motivation

In nature we find groups of agents that cooperate to solve complex problems in highly distributed ways [28]. The best known examples include colonies of ants, bees and termites. Through various techniques, these biological systems manage to accomplish colony level tasks, despite being comprised of primitive agents without strict centralized control. Cooperatively these groups meet their own challenges efficiently using a rudimentary but robust form of communication. The notion that this form of problem solving emerges from the collective actions of the simple colony members has been termed *swarm intelligence* [33].

Biologists realize that the physical channel by which agents in these systems communicate is the environment they live in. Depending on the species and task, it seems agents are capable of both reading signals from and writing information to their surroundings. That is, they may use environmental cues when deciding their next action, and they may instinctively leave new physical markers based on their experience. This special form of indirect communication through a physical medium is called *stigmergy* [29]. The principle of stigmergy is used to broadcast various messages and instructions in natural systems. In the following we list some examples.

- *Recruitment for cooperative work.* The existence of a particular type or size of object in the environment may induce agents to begin acting in a cooperative

fashion. Ants instinctively congregate, for example, around a large resource and together manage to carry the object.

- *Tasks governed by spatial awareness.* The current state of the environment may be used to decide how some spatially oriented task proceeds. Consider, for instance, how ants have been observed to sort their brood. Each ant picks up a larvae and drops it next to other larvae of the same size. The process repeats until all the larvae are sorted.

- *Path selection through pheromone trail laying.* Using a chemical marker called pheromone, systems of ants can determine best paths to travel. The results emerge provided each ant deposits pheromone as it moves, and simultaneously exhibits a preference for trails with higher concentrations of existing pheromone. Since more ants can travel a shorter path in a fixed amount of time, better routes receive more pheromone. At the same time, deposited pheromone evaporates after a time so that paths must receive a continual reinforcement of pheromone to be considered the desirable solution. This mechanism even allows different routes to be preferred at different times as appropriate.

Although theoretically we can learn from any of these natural behaviours, by far the most widely exploited mechanism is the notion of path selection based on pheromone reinforcement. This idea alone has led to an entire class of algorithms, aptly named *ant algorithms* [19], which are capable of addressing various types of optimization and control problems. In the next section we will see in more detail how depositing artificial pheromone in an artificial environment can be used for problem solving.

1.2 Ant Algorithms Described

The best way to understand the principle underlying ant algorithms is through a concrete example. Let us take a closer look at an idealized system of real agents. Suppose a colony of ants wishes to decide between 2 well defined routes from their nest to a single food source. Moreover, assume that any number of ants can be on either path at any time without interfering with one another. Imagine that there is a large enough pool of these agents foraging for food to allow the rate of pheromone build up to overcome the rate of evaporation. Further, assume that ants travel at essentially a constant rate, and that one route is physically twice as long as the other. It should be clear that the shorter route is the more efficient solution under these conditions. Finally, suppose that each ant is genetically programmed to favour paths that have a higher concentration of pheromone, and that they provide feedback about which path they select by adding to the pheromone level on the route they use.

Even if initially there is no difference between the routes in terms of their respective pheromone levels, the shortest path will quickly emerge as the desired route. This occurs because the rate at which pheromone builds up on the shorter route initially out paces the rate it builds up on the longer route. The cause of this discrepancy in rates is a result of ants taking less time on the shorter path, causing a larger rate of positive reinforcement to the shorter route.

Some key points to note exist even in this small motivating example. We observe that implicit in the setup of the problem there is a graph structure. Here it is a simple graph with only 2 nodes, the nest and the food source, and only 2 links, the 2 single hop paths between them. The nodes correspond to choice points, and we assume

that the relative amounts of pheromone on outgoing links can be observed at each node. Next, there is a cost associated with traversing a link. In our example the fixed cost was a result of the constant rate of travel, combined with the constant physical length of the paths. More generally, overall cost could vary depending on a variety of factors. Finally, the goal is to determine the shortest path in terms of cost, and then leverage this information to do useful work. In this particular example, the goal is to simply route the ants. However, in general the relative amounts of pheromone can be used to bias other decisions as well.

Many optimization problems can be viewed as finding a least cost sequence of choices in a network of choice points. Perhaps the most obvious, and a motivating application that we consider heavily in this project, is that of determining the best routes to use in a communication network. For this reason, we will see that ant algorithms can provide elegant and highly effective solutions to these sorts of problems.

So far, we have only described the principle used by real ants. Our goal at this point however, is to understand how to map this biological metephor onto an artificial system that represents a problem to be solved. To this end, in the following we attempt to define and describe the standard features of a generic ant algorithm.

As we have suggested, the basic structure of the artificial system is a graph. The nodes of this graph represent all the points at which choices can be made as we attempt to construct a feasible solution. The links reflect the outcome of choices in some sense. They direct the flow from one decision to another, and account for all costs associated with making a given choice. This graph is like the environment in which our artificial ants live. Any ant algorithm will have an associated graph. Furthermore, for any problem in which an ant algorithm is a viable solution method, we must be able to

interpret the objective as finding an optimal cost path in this graph structure.

The ants themselves can be pictured as a collection of autonomous mobile agents which are allowed to move from node to node in the environment by making decisions and traversing the links. As these ants travel they either experience actual link costs, as is the case in the idealized real ant system above, or they are sophisticated enough to simply measure and record them. This process of information gathering is commonly termed the *forward* function of ants. Once they have built a complete solution, indicated by reaching a particular destination node in the graph, they release feedback about their journey. The act of depositing feedback is typically called the *backward* function of ants. The actual feedback information is like the pheromone of a real ant system, and it is placed so that future ants can make use of it. It accumulates at the choice points in which it pertains. Note that this in unlike real ants which deposit pheromone continuously. In an artificial system, decisions are only made at the nodes, and so it is unnecessary to have pheromone on the links themselves. We must also bear in mind that artificial ants are capable of depositing varying amounts of pheromone to reflect the goodness of the solution they have found. This too is different from real ant behaviour. Each node contains a structure called a pheromone table which houses the deposited feedback. A given row of this table corresponds to a specific destination node that is reachable from the choice point in which the table resides. The row contains all the pheromone levels of the possible first links on a path to a destination node. Pheromone tables are a critical ingredient of ant algorithms. They represent the collective memory of the ant system by storing the experience of many previous ant probes. This shared memory enables the learning mechanism of ant algorithms.

Another important aspect of both natural and artificial systems is the evaporation or reduction of pheromone over time. This process can be mimicked in ant algorithms by simply ensuring that pheromone table entries are periodically reduced in some fashion. The most common approach is to insist that each time a row of a pheromone table is updated by increasing one entry, as a result of some new feedback, the row is renormalized such that its elements sum to one. This has the effect of reducing the previous values of every entry in the row. As a benificial side-effect, having normalized pheromone values in a row makes it easy to view the row as a probability distribution when ants need to select a path from their current position to a particular destination node.

Finally, it is important to note that the ultimate solution arising from an ant algorithm is really a function of the evolved pheromone tables in general. This fact is masked in the description of the working of real ants. In that case, the goal is to deduce efficient routes for the ants themselves. Since the ants are already following the pheromone tables that they maintain, this task is being performed implicitly. In more complex artificial problems however, ants simply build pheromone tables from which we can deduce a result. That is, the best solution to the problem at any point is typically attained by viewing the pheromone tables in a specific way. Furthermore, the current solution, as a function of the current pheromone table entries, may even affect how the tables evolve in the future. We will say more about this type of situation later. For now it suffices to note that with ant algorithms, unlike real ants, our interest often lies with another process that might be better described as a consequence of the exploration and feedback gathering of the ant-like agents.

Although all ant algorithms incorporate the basic principles outlined above, there

can exist significant variations in the way they are implemented. Usually operational choices are dictated to a large degree by the problem application. For example, we now list the major considerations that must be kept in mind while defining an ant algorithm for a particular purpose.

- *Link cost variability.* It is critical to know whether or not the link costs are static or dynamic. In certain situations, such as the real ants foraging for food, it is sufficient and appropriate that the link costs are fixed. At other times however, it is important to realize that the costs fluctuate over time. This is the case with communication network routing, for example. In that application the actual delay crossing a link can be negligible, when compared with the processing and queueing delay that data packets experience. Note that the cause of the dynamic link costs may or may not be related to the operation of the ant procedure. At any rate, the importance of this distinction cannot be under emphasized, because it pertains directly to the goal of the algorithm. In the first type, where the costs are fixed, the goal is to find the optimal solution in the least possible time. In the second type, where the associated costs are changing with time, the point is usually to track the changes as accurately as possible and remain a reasonable distance from the instantaneously optimal solution at any given time. This latter type is more about control of the dynamic situation than optimization of a static problem.

- *Algorithm distribution.* Another crucial design question is whether the algorithm has a centralized or decentralized focus. A centralized ant algorithm runs with knowledge of the underlying topology and individual link costs. Moreover, there typically exists a central clock, and each algorithm step the current

pheromone levels are used by a set of ants that each select a entire potential solution path. Pheromone table updates are applied after all the solutions in a given step are analyzed. In stark contrast, in a decentralized implementation of an ant algorithm there is no global synchronization of events, and no central entity that performs calculations over the whole graph. In this form, individual ants are simply dispatched at regular intervals. They work autonomously to build solution paths and produce feedback. The pheromone tables evolve based on updates triggered by feedback as it appears, and a preferred solution emerges.

We note that static optimization problems are usually tackled with a centralized version of an ant algorithm. This follows from the fact that solutions to static problems can typically be calculated off-line, with complete knowledge of the underlying system. On the other hand, ant algorithms for dynamic control often require a level of decentralization because of the changing nature of the problem. It is easier to let agents adapt to their local situation autonomously, than try to have a single authority oversee all actions. A distributed ant algorithm is ideal for this type of on-line or real-time situation. We should remark however, that there exist static link cost applications in which a decentralized algorithm is desirable. Similarly, certain dynamic link cost problems might be best solved using an ant algorithm with partially centralized functions.

There are also several minor variations in ant algorithm procedures that deserve some mention. A factor worth thinking about is the generation of feedback information. It must be decided, for instance, what actions trigger feedback as an ant travels. The simplest idea is to just give feedback when a complete solution path is constructed,

but it is also possible to generate feedback at intermediate points along the way as well. Also, feedback has to be incorporated into pheromone tables to be useful. The number and location of tables that are updated for each piece of feedback gathered must also be determined. Other potential alterations to consider pertain to enhancing the ability to explore and adapt solutions as necessary. Because ant algorithms search for the optimal solution, any modification that increases the rate of exploration, while still allowing good solutions to settle out when appropriate, is highly desirable. Most modern ant algorithms include some form of enhancing feature that encourages discovery of alternate solutions.

The real beauty of ant algorithms is the simplicity with which they can be implemented and managed. Once the graph structure is established and the artificial agents are programmed, there is no further need for centralized control. The algorithm can be set in motion and very quickly it yields reasonable solutions. Furthermore, ant algorithms are resilient to small errors. Just as a colony of real ants is not compromised by the loss of a few agents, a small number of corrupted pieces of feedback likewise does not significantly alter an entire pheromone table. Additionally, by including evaporation, an ant algorithm is quite suited to dynamic situations in which the optimal solution is a constantly moving target. Finally, because the critical interactions take place at the local level, and there is no need for centralized control, scaling ant algorithms for use in systems with many nodes and links is usually feasible.

1.3 Feedback Modelled By Urns

A main goal of this research is to capture the essence of ant algorithms using mathematical models. To accomplish this, we need to understand the inherent stochastic search and feedback process. We must evaluate the sources of any randomness to appreciate their effect on the evolution of the algorithm and the final result. This can be particularly complicated in an ant algorithm with dynamic link costs that depend on the actions of ants. For instance, consider an ant-based network routing application. The congestion level of data traffic determines delays in the network. But the delays are taken to be the link costs, so ants from the routing algorithm both affect and are affected by the congestion that is present. There are two related stochastic processes to track in these cases. One process represents the changing pheromone levels and the other reflects the changing link costs. Both must be taken into account in order to model this type of situation properly.

In light of the above discussion, initially we restrict our view to ant algorithms with static link costs. This allows us to focus our investigation exclusively on the evolution of pheromone values. We observe that a key feature of these ant algorithms still remains the notion of self reinforcement and convergence. The more often a solution path is tried, the more likely it will be tried again in the future. Moreover, as time goes on the algorithm typically favours a small set of better solutions over the others. We say typically since, at least in the case of real ants, there is a chance that a suboptimal solution is reinforced to the point that it becomes self sustaining [28].

For good reason perhaps, given the original inspiration for ant algorithms, this behaviour is like a biological evolutionary process. A probabilistic model that works

much the same way is the Pólya urn, proposed by Eggenberger and Pólya in 1923 [22]. A simple incarnation is described as follows.

Suppose an urn contains some number of white balls and some number of black balls. At each stage we draw a ball randomly from the urn and then replace it, together with another ball of the same colour. The process is repeated. Clearly, as the number of balls of a given colour increases, so does the probability of drawing that colour. Eventually, and depending strongly on the initial numbers of each colour, one colour of ball may dominate the composition of the urn. This occurs by the same mechanism as in ant algorithms. Present choices affect future ones by bringing feedback to the system that hedges the evolution one way or another. The contents of the urn store information regarding the previous sequence of draws. Future draws are based on this history in a fashion similar to the way a pheromone table tracks feedback that influences later decisions.

This simple example is essentially the original process. But urn models have been around for some time, and many generalizations exist [36]. For our purposes, at least a couple of abstractions are crucial. First, it is possible to consider more than 2 colours in an urn. Since we intend for the colour types to correspond to different outgoing links from a decision point in an ant algorithm graph, we require this potential. Second, the number of balls returned for a given colour type drawn can be any predetermined value. This is a critical enhancement, with respect to ant algorithm modelling. We want to be able to differentiate the amount of pheromone rewarded. A outgoing link associated with a path with less cost should receive more pheromone. We can only do this within the context of an urn model if we can add differing numbers of balls depending on the colour type selected.

It seems that a Pólya urn has the potential to be a good model for the evolution of an ant algorithm. The only problem lies in the fact that this standard urn does not include a mechanism like evaporation. As we have described it, the number of balls in the urn actually grows to infinity as the process continues. The proportions of balls may stabilize, but the equilibrium amounts may be such that it would be nearly impossible to draw the inferior colour again. Moreover, technically there is a trace contribution from the outcome of every draw since the origin of time, but the effect of additional balls significantly diminishes as time goes on. This is not like an ant-based procedure. In an ant algorithm, evaporation ensures that a solution must be continually reinforced in order to stand out. Recall that this allows new solutions to take hold when necessary.

Luckily, a variation of the original Pólya urn exists that only allows balls to remain in the urn for a fixed number of draws [1]. We will see that this finite urn is a better starting point for our analysis.

1.4 Thesis Contributions

Ant algorithms inspired by biological systems provide a novel method for solving certain difficult problems. Because they include random elements though, verifying the utility of an ant algorithm involves much more than just proving its correctness. Ant-based methods should not be measured in absolutes, rather, we must evaluate the probability of them behaving in a desired way. For this reason, studying them using a mathematical foundation is worthwhile and fruitful.

In this work we set out to study ant algorithms using probabilistic methods. As we

have already suggested, our launching point is a simple finite urn model. Prior to this project, urns with fixed memory had not been investigated much beyond their introductory appearance. Thus, our initial efforts extend the concept of finite urns in several dimensions. We define them in a general framework, which we term a functional weight finite urn, that allows many colour types and extremely flexible update procedures. In this general setting we verify a closed form stationary distribution, and derive expressions for the stationary mean and covariance of the number of balls of each colour in the urn. We also demonstrate the usefulness of the model by analyzing various applications. In particular, we use our generalized finite urn to investigate an ant algorithm with static link costs.

This initial model is then studied further as the fixed memory size increases. We derive limiting scaled mean and covariance expressions for the proportions of the number of balls of each colour in the urn. In doing so, we show that this limiting behaviour of a finite urn is similar, but not identical, to the time limiting behaviour of an analogous infinite urn. Specifically, we prove that the limiting scaled composition of a sequence of stationary finite urns indexed by the memory size has the same mean vector but a different covariance matrix compared with the known limiting scaled composition of a comparable standard Pólya urn.

We also look at an ant routing algorithm in greater detail. This application is interesting because it highlights many of the strengths of ant algorithms. For instance, it is a dynamic link cost problem where the goal is efficient control of a fluid system. Moreover, it is a case in which it is desirable to have the algorithm run without centralized control.

Our functional weight finite urn is not suited to modelling this dynamic situation.

Hence, it is extended to include a component that can portray the randomness in the link costs. We call this new Markovian process an embedded functional weight finite urn, because of its ability to model a sequence of draws from an urn at a given choice point, under the influence of the outcomes of similar processes at every other node in a decision network. After calculating a stationary distribution using standard methods, we derive interesting gauges of the interaction between the sequence of draws of this urn and the driving external process. We then use these measures to study both steady state and transient behaviour of the dynamic link cost and distributed ant routing problem.

The knowledge gathered through our analytic modelling is eventually harnessed to construct an alternative ant algorithm framework. The new paradigm, which applies to the same set of problems as the original ant algorithm method, provides greater control over the way feedback is utilized. Specifically, we move the collective memory of the ants from the traditional pheromone table to a database. We then build any number of pheromone tables from the database using update functions that are tailored to the particular use of each table. Ultimately, this decouples the exploratory work of ants from any other subsystem that wishes to make use of a pheromone table like structure. The approach is shown to be highly effective for the dynamic ant routing application using a computer simulation analysis.

These contributions are developed in approximately the order they are given above. More will be said about their explicit location within this document in the next section.

1.5 Thesis Outline

Immediately following this introduction, we will review related literature in Chapter 2. This step is crucial, as it summarizes the existing fields of ant algorithms and urn models. Once this survey is complete, the next 4 chapters of this document develop our main results.

Beginning with Chapter 3, we describe our first ant algorithm model: a functional weight finite urn. We address its time limiting stationary distribution, and other first and second order steady state quantities like means and covariances of the number of balls of each colour type in the urn. Moreover, this first main chapter also presents 2 key examples. One is an application in the field of Information Theory, the original context in which finite urns were proposed. The other is a static link cost ant algorithm.

Chapter 4 continues to explore our functional weight finite urn. This time our emphasis is on its behaviour as the memory size increases. Essentially, we extend the results of Chapter 3 to their limit in this spatial dimension. Chapter 4 also formally characterizes the similarities and differences of a limiting scaled functional weight finite urn, as compared with the limiting behaviour of an analogous infinite urn. We then conclude the chapter by conjecturing the distributional form of the limiting scaled composition of a finite urn.

In Chapter 5 we return to our study of ant algorithms, and develop an extended process which we call an embedded functional weight finite urn. This second model, which can be thought of as a pure generalization of the functional weight finite urn, is more appropriately suited to the task of modelling ant algorithms with dynamic link

costs. Specifically, this new Markovian process accounts for the partially dependent and dynamically changing link costs that both drive and are driven by the operation of ants. Unfortunately, fewer closed form results are available for this more sophisticated model. Still, this chapter sees the design of measures that effectively gauge the control asserted by an ant algorithm for the class of dynamic link cost problems. Indeed, the focal point at this stage is clearly our key ant-based routing application, whose performance we analyze thoroughly using the developed theory.

Furthermore, applied contributions are given in Chapter 6. There we switch modes and propose improvements to existing ant algorithms based on our analytic modelling in Chapters 3 and 5. In fact, we present a completely different paradigm for ant algorithms in this chapter. In particular, we introduce the concepts of database enabled memory, and distinct pheromone tables for distinct purposes. Once again the bulk of Chapter 6 looks at the ant routing example. Through a computer simulation of this application we confirm the benefits of our new ant-based framework.

Finally, some concluding remarks are made in Chapter 7. We give a summary discussion of all the results presented, and we also comment on the direction of our continued research.

Chapter 2

Literature Review

An applied mathematics problem requires a substantial survey of background material. A basic introduction to the problem area itself and any previous analytic attempts to study it are in order. Furthermore, all relevant theory corresponding to the intended mathematical approach must be discussed. In the case of this work, for example, we want to address the field of ant algorithms, including their biological origins, and examine any existing models for ant-based solution methods. We also want to canvass known results regarding urns, since we will develop related probabilistic tools as part of this research.

2.1 From Natural to Artificial Systems

Without specific biological studies, ant-based algorithms would not be prevalent today. By understanding the principles underlying colonies of ants, bees and termites, for instance, biologists contribute the fundamental ideas behind paradigms based on

natural metaphors. Working with swarms of social insects, they have painstakingly deciphered and catalogued the subtle methods of communication that lead to cooperation amoung the members.

Everyday experience tells us that groups of simple interacting agents can work together in a distributed fashion to accomplish some global task. Surprisingly, it is also fairly obvious that sophisticated behaviour can arise without any clear centralized control. This notion, which is of primary interest in [33], [50] and [7], is termed swarm intelligence. It is the key attribute underlying ant algorithms.

With respect to a collection of organisms, global behaviour can only arise through effective communication. Grassé introduced the term stigmergy to describe indirect communication between agents through their environment [29]. The author realized that even simple agents must communicate with one another, and proposed that they do so by maintaining specific markers in their shared habitat. For instance, it is now well accepted that ants in particular, make extensive use of chemical substances called pheromones to share information with one another. Through laying pheromone and simply being aware of it, ants can collectively exhibit quite sophisticated behaviour such as discovering optimal paths in their environment.

Ants are possibly the earth's most organized species. Goss and Deneubourg et al. have conducted some famous experiments with pheromone trail laying ants. Some of their crucial findings are reported in [28] and [15]. Part of their work involves elegant bridge experiments that demonstrate how ants determine efficient paths. The apparatus used in the investigation is a clever design. It consists of two trays where ants can pool, and two bridges between them. The bridges, which are of different lengths, constrain the ants in their travels. The ants are not allowed to wander

aimlessly. They must commit to one of the bridges in order to pass from one tray to the other, which they are enticed to do with food as part of the experiment. The outcome is that while initially the ants choose either bridge with approximately the same frequency, after a short time the majority of the ants are using the shorter route. The authors conclude that the ants lay pheromone as they travel, and that the increased concentration of pheromone that builds up on the shorter route, because more ants are able to traverse the short route, signals the optimal choice. In [28] the authors even present a differential equation model for this mechanism. The system of equations can not be solved in closed form, but, the numerical results agree well with observations of the real ant system. Other authors have confirmed similar ant behaviour through experiment and analysis. For example, [48] conducts a simulation study of ants that attempts to better understand the notion of laying trails and following them.

Certainly the basic concepts upon which ant algorithms are designed stem from nature. Still, it takes human ingenuity to harness these ideas and build useful systems. Colorni, Dorigo and Maniezzo are credited with the first attempt at leveraging concepts from natural systems to solve optimization problems [10], [11], [17]. Good summaries of the biological inspiration behind their original system, and modern ant algorithms alike, are presented in [7], [8] and [20]. Much more will be said about ant algorithms in the next section.

It is worth noting that ant algorithms are not the only artificial systems built using a biological metaphor. In fact, some would argue that they are just one type of agent system. To put ant-based procedures in perspective, it is worthwhile considering related literature. A nice summary of the terminology surrounding agent systems is

presented in [24]. The paper classifies agents according to various attributes they may posses. Some of the more common traits are the ability to be autonomous, communicable, mobile and adaptive. To complicate matters, the term mobile agent is often used to refer to agents with much more general capabilities. That is, there is an entire body of work dedicated to mobile agents alone. Thankfully, [43] gives us an overview of this particular area.

2.2 Ant Algorithm Applications

The field of ant algorithms is just over a decade old. In this short time however, it has flourished. To date, ant algorithms have been proposed for many types of problems. Broadly, these fall into 2 main types. The first class is characterized by static link costs. The goal is almost certainly discrete optimization in this case [19], [20]. A common feature of this category is that the algorithm is usually implemented with some degree of centralization. Typically they include steps where solutions are compared before updates are applied. The second class is reserved for applications with dynamic link costs. In these problems there is no single optimal solution. Effectively, the problem is always changing. The objective of the ant algorithm then is to closely track the changes and manage related processes. Often it is a requirement in dynamic control problems that the algorithm is capable of running completely decentralized.

It is impossible to detail every problem that has been attempted using an ant-based approach. Instead, in the following sections we will merely outline some of the notable uses.

2.2.1 Static Optimization Problems

Colorni, Dorigo and Maniezzo were some of the first authors to use the ant paradigm. Their original algorithm, called Ant System, is discussed in [10], [11] and [17]. It is used in this early work mainly to study the travelling salesman problem. This hard optimization problem is an ideal test case. It is very clearly concerned with finding an optimal cost path in a network structure. Moreover, its difficulty is well known, making it an excellent benchmark for optimization algorithms. We note that the original Ant System is not the best algorithmic solution to the travelling salesman problem, but it is nonetheless a very novel approach with much potential, and this is the reason it has received so much attention from the moment it appeared. From our point of view, perhaps the most interesting part of [10] is the included future work section at the end. The very first of four main research directions listed calls for the formulation of mathematical theory to describe the exhibited behaviour of ant methods.

Many other examples of ant algorithms have emerged since Ant System. In [46] the problem of virtual wavelength path routing and wavelength allocation is considered. [9] considers a generalized version of the travelling salesman problem called the probabilistic travelling salesman problem.

Additionally, researchers have continually advocated enhancements to the basic ant methodology. Many modifications come at the hands of original authors, as in [26] and [18]. At times the algorithms are given new names to reflect the changes. Ant-Q, for instance, appears in [26]. Ant-Q is essentially a generalization of the original Ant System that specifies a framework for ant algorithms that recognizes the similarities between them and a technique called Q-learning. More importantly, the modifications

that lead to Ant-Q make it one of the first ant algorithms to be truly competitive with other specialized algorithms designed to tackle the asymmetric travelling salesman problem.

As the field matures, significant contributions are made by other authors as well. Stützle, for instance, suggests an important extension to Ant System in [44]. This variant, called MAX-MIN Ant System, has received a lot of attention because it out performs the original by only allowing pheromone updates that are derived from the best solutions at each step of the algorithm. This has been shown to be an effective technique under the right conditions. To compliment this more aggressive searching mechanism, which on its own would have a tendency to converge quickly to any solution, optimal or not, a provision is included that bounds the pheromone levels between some maximum and minimum values at all times. These upper and lower restrictions ensure that good solution components, indicated by high pheromone levels, are never regarded as too good, while at the same time bad solution componentss, normally indicated by low pheromone levels, are never regarded as too bad. Ultimately the stochastic searching process is drawn out where it might otherwise converge too rapidly. This concept of forcing pheromone levels to respect both an upper and lower barrier, often referred to as a ceiling and floor respectively, is a technique borrowed by many researchers. In [44] the authors of MAX-MIN Ant System claim that their algorithm, as applied to the travelling salesman problem and the quadratic assignment problem, performs at least as well as other cutting edge algorithms for those problems at the time. They back this claim through comparisons with algorithms based specifically on tabu search methods.

Although there exist differences in implementation, all ant algorithms follow common

principles. Because of this fact, Dorigo and others have successfully outlined an Ant Colony Optimization meta-heuristic that links a large class of ant algorithms based on their standard features [19], [20]. The unified framework is even given special treatment in a text covering recent developments in the field of optimization [12].

2.2.2 Dynamic Control Problems

In the beginning, problems with static link costs were the focus of ant algorithms. Indeed, it is easy to think of the motivating real ant system as having fixed costs for the various alternate paths. Arguably the real power of ant-based methods however, is in the ease with which they can apply to environments with dynamically changing link costs. By far, the driving application motivating study of these more sophisticated types of problems is communication network routing. In this problem, at any given time we would like to have a set of optimal routes, with respect to current data traffic congestion, to be used for transmitting information between any pair of network nodes. This problem is particularly complicated by the fact that the very link costs that the algorithm must track are in turn affected by and fluctuate as a result of the decisions of the algorithm.

Several authors suggest using ant methods for the network routing problem around the same time. Typically the implementations have a similar setup. At each node there are pheromone routing tables. These tables are similar to the next-hop routing tables used in other algorithmic approaches. That is, each row of the table at a given node corresponds to a possible network destination node, and the entries by column suggest a probability of using the corresponding node as the next-hop. In an ant-based implementation the pheromone routing tables are maintained by the functioning of

ants. Ants flow in the network, using the routing tables to guide them, and gather information about specific paths. They then use this gathered feedback to update appropriate table entries. Simultaneously, data flows in the network using the same routing tables. The goal is to route data efficiently, and it is really a consequence of the ants routing themselves efficiently.

The first ant routing scheme, proposed by Schoonderwoerd et al., appears in [40]. The authors demonstrate several highly desirable characteristics of ant-based procedures in their work. They suggest a relatively simple distributed algorithm for the dynamic link cost problem. They note that their solution is robust, adaptable, scalable and yields a routing policy with good load balancing properties. One shortcoming of this implementation is that it assumes that the flow of traffic in one direction is always similar to the flow in the opposite direction.

Another version of an ant inspired network routing algorithm that has received a lot of attention is called AntNet [16]. This implementation by Di Caro and Dorigo is notably more complex than the Schoonderwoerd et al. proposal. The advantage is that it is more realistic. It is assumed that links, and hence entire routes, in a communication network are directed. Feedback gathered by travelling in one direction is used to apply pheromone updates for future traffic, ants or data, wishing to move in the same direction. To facilitate this notion, in AntNet the authors define 2 types of ants. Initially, a *forward ant* gathers feedback by using the next-hop tables to build a route in a specific direction from a source node to a destination node. Then, when a forward functioning ant reaches its destination node, a *backward ant* is created with the sole purpose of transporting the feedback, using the same route but in the opposite direction, to the original source node. Along the way the backward ant triggers any

updates to the appropriate pheromone tables. Having 2 types of ants in the network increases the network routing protocol overhead, but at least the collected feedback is used accurately.

The above implementations are not the only contenders in this area. Other ant-based routing algorithms for telecommunication networks have been put forth. For example, consider the efforts in [6] and [32]. Without going into detailed descriptions of each, it suffices to say that when compared to traditional shortest path type routing algorithms such as the distributed Bellman-Ford algorithm or Dijkstra's algorithm [5], ant routing algorithms perform well. This is especially true if characteristics like load balancing ability, fault tolerance, adaptability and scalability are included as part of the evaluation criteria.

Finally, it should be pointed out that technically both static optimization problems and dynamic control problems can be stated in the Ant Colony Optimization framework [19], [20]. As new applications arise, even those with dynamic link costs, they are frequently claimed to fall under the Ant Colony Optimization meta-heuristic umbrella. This has the potential to be misleading. In the case of dynamic control problems, the primary role of the ant-based algorithm is typically not perceived as optimization, but management of an evolving system.

2.3 Ant Algorithm Modelling

The majority of research on ant algorithms is conducted via computer simulation. At this point, few analytic results regarding ant-based systems exist. There are a couple of exceptions however. In this section we will briefly describe some of the

mathematical studies of ant algorithms.

One interesting class of results are convergence proofs. Recall that ant-based methods are a type of stochastic search in which we take the solution to be the one that emerges as a result of the highest level of pheromone reinforcement. That is, we hope that the algorithm converges to the correct answer to the problem under consideration. In most practical implementations however, it is possible for the algorithm to converge to a locally optimal point in the solution space that is not the globally optimal value. This undesirable behaviour is even present in natural ant systems [28]. However, for certain idealized implementations, it is possible to prove that an ant-based algorithm will eventually find the global optimum.

Gutjahr has developed convergence results in [30] and [31] for specific static link cost ant algorithms. The proofs rely on the assumption that only the best solutions get reinforcement at each step of the algorithm. This constraint, which implicitly assumes that the procedure is also centralized, is known as an *elitist strategy*. In both [30] and [31] the ant-based algorithm under consideration is framed as a Markov process. The limiting results are consequences of limiting Markov theory. We observe that the analysis does not address the rate at which the convergence takes place.

In [45], Stützle and Dorigo put together a direct probability argument that can be used to justify the optimal convergence of a particular class of Ant Colony Optimization methods. At its heart, the work uses the fact that a non-decreasing and bounded sequence converges. As with Gutjahr's work however, the result does not say anything about the rate of convergence. In practice, the usefulness of any algorithm for a hard optimization problem will be scrutinized in terms of its speed. Knowing more about the rate of convergence would be an ideal situation.

Another class of analytic models for ant algorithms focus on understanding the underlying mechanisms of the natural paradigm. One of the first papers appearing in the literature with this type of goal is by Merkle and Middendorf [37]. These authors also consider static optimization problems to be the target of their ant-based procedures. The system they model is assumed to be highly centralized. In fact, their analysis imagines that at each step the update to pheromone tables is the expected update from an infinite number of iterations conducted with the current environment fixed. It is a unique and interesting approach. Note that their goal is not to show convergence of a particular type of ant algorithm, rather, they illuminate the relationship between the use of feedback and the evolving decisions of ants.

More recently, a pair of researchers have undertaken to model a dynamic link cost application of ant-based methods. In [13] and [4] Costa and Bean tackle the analysis of a distributed ant routing algorithm. Their procedure is straightforward, and quite comprehensive. They manage to construct an elaborate iterative system of equations involving quantities representing the network pheromone tables, the ant routing probabilites and the data routing probabilities. Each of these quantities is a function of a discrete time index. The evolution of an ant algorithm is modelled as the process of iterating these 3 main values, which are dependent on each other. More specifically, at each stage the current pheromone table values dictate both the ant routing probabilities and data routing probabilities. But the current pair of routing probabilities can be used to calculate new pheromone table values. New pheromone table values then specify modified routing probabilities and the process is repeated over and over. Costa and Bean essentially prove that as long as the data loading scheme is light enough, this iterative process actually converges to a fixed point in the sense

that each of the pheromone table values, the ant routing probabilities and the data routing probabilities settle down to specific unchanging quantities. These fixed values can be used to evaluate the equilibrium performance of the emergent routing policy.

Perhaps the most interesting part of the work by Costa and Bean is that they manage to suggest ways to directly improve the design of an ant routing implementation based on the findings of their analytic model. Specifically, they show that one way of increasing equilibrium performance, based on the measures they consider, is to place particular restrictions on the choice of routing probabilities as functions of the pheromone table values. Typically, a data packet simply routes itself using the single best known path indicated in the appropriate pheromone table. This makes sense as there is no need for data to test alternate routes. Meanwhile, ants are usually encouraged to explore each of the alternative paths by choosing their next-hop link in such a way that they are biased toward the various possibilities in proportion to the respective pheromone table values. In [4] however, it is proposed that data make decisions as before, and ants also choose their path without exploration except for the first hop. This new policy is shown to have better equilibrium performance. The authors suggest that the reason this works well is that the experience of the ants is more in line with the actual experience of the data in this case. Without the aid of their analytic model though, it seems fair to say that this modification is quite non-intuitive, and hence would be hard to discover by other means.

2.4 Pólya Urn Models and Generalizations

In Chapter 1 we briefly outlined our intention to use urn processes as a basis for modelling ant algorithms mathematically. With this direction in mind, here we take the time to understand some relevant urn facts.

Pólya urns have existed as flexible probabilistic models for almost a century. Initially they were introduced as a way to model the spread of disease within a population [22]. The most basic formulation consists of an urn in which balls of only 2 colours, typically black and white, can exist. Starting off with some number of each colour, say b_0 and w_0 for black and white respectively, at each time step we draw a ball from the urn, note its colour and replace it together with another ball of the same colour. This procedure is repeated indefinitely, with no balls ever removed. Note that for our purposes we will often refer to this type of process, where the total number of balls grows without bound, as an infinite urn. Usually the evolving composition of the urn over time is of central importance. Here we take composition to denote the vector of the number of balls of each colour type.

Infinite urns have proved to be very versatile mathematical tools. In fact, entire texts have be dedicated to describing various types of urns and their applications [36]. Thanks to the various generalizations of the basic urn put forth by Eggenberger and Pólya they have been used to successfully model many kinds of random phenomenon. Some typical extensions we find in the urn literature are given below [36].

- Arbitrary numbers of colour types.
- Number of replaced balls different from 1.

- Arbitrary amount of balls of any colour type may be added for a given colour type drawn.
- Balls may be removed, as opposed to added, during update.
- Random number of balls replaced.

Of course various combinations of these have also been considered. By now it is common practise to characterize an urn with deterministic replacements by a ball addition matrix [36]. In row i and column j this structure lists the number of balls of colour type i that are replaced, in addition to the one drawn, when colour type j is selected from the urn. In this research it will be important to be able to discuss very general ball addition matrices.

One aspect of infinite urns that receives continuing development is that of finding the limiting distribution of the scaled composition. Since time is the standard scaling parameter, the scaled composition can be interpreted as the vector of the relative proportions of each colour type in the urn. In the simple 2 colour type case where the only balls added each draw are a fixed number of the colour types drawn, it can even be shown that in the limit the distribution of the the scaled number of white balls (or black balls if we prefer) is Beta with parameters that are a function of the initial weights b_0 and w_0 [22], [36]. More generally a C colour type case where the only balls added each draw are a fixed number of the colour types drawn, it can be shown that the limiting distribution of the scaled composition is Dirichlet of order C with parameters that are functions of the initial number of balls of each colour type [21], [36].

Along these same lines, limit theorems have been developed independently for a number of other special case ball replacement schemes. In a couple of relatively new papers, Janson provides a treatment of limit theorems for a number of classes of infinite urns with very general replacement policies [34], [35]. The combined body of research presented in these 2 papers is diverse enough to recover many of the limiting results put forth by other authors in the area. In this sense, one the greatest contributions of [34] and [35] is the unification of the previously ad hoc collection of results into one theory. It must be noted however, that Janson also extends previously known results in many ways. At any rate, when comparisons of limiting behaviour are warranted later in this work, we will attempt to draw parallels to Janson's results because of their completeness.

An extremely appealing variation on traditional Pólya urns is proposed by Alajaji in [1]. In the new type, the drawing and replacement process is the same as for infinite urns. The key modification is that all balls, except any initial ones, may only remain in the urn for a fixed number of draws after they are added. Because of this important distinction, this type of urn is referred to as a finite urn.

Initially only the analogue of the basic infinite Pólya urn has been considered in [1]. That is, only 2 colours and constant sized replacements of the same colour type drawn are discussed. However, finite urns can be thought to parallel original infinite urns in the sense that all the extensions mentioned above in the context of regular urns, can be at least contemplated now in the context of finite urns. In particular, there is no problem characterizing finite urns by a ball addition matrix with the same interpretation in terms of replacements.

The really interesting consequence of a finite urn model is that it is Markovian. This

allows analysis that is not possible in a corresponding infinite urn. We will use this observation to our advantage in the next chapter as we describe general finite urn and later analyze it.

Chapter 3

A Generalized Finite Urn

In this chapter we present a model with the purpose of analyzing some basic ant algorithm behaviour. We define a specific finite urn for our purposes. Recall, finiteness in this context refers to the fact that balls may only remain in the urn for at most a finite amount of time. The urn is quite general in the sense that it may contain balls of an arbitrary number of colours, and at each step, balls of any colour may be removed or added. The actual number of each type will depend on the colour drawn and the current state of the urn. In terms of the possible extensions mentioned in the Literature Review chapter, this is a very general urn. The only extension not considered here is that of a random number of balls returned. That is, we assume that the number and type of balls returned for a given type drawn and state are determined.

To facilitate tracking balls based on their age, as measured from the time they were added, we imagine the urn is divided into a sliding window of slots. That is, the urn behaves like a first in, first out queue with respect to the addition or removal of balls.

A set of balls added at one point in time are placed in a slot and removed a specified length of time later. More will be said about the update procedure in the following.

We have already seen that urns are a good starting point for modelling ant algorithms. Their basic operation accounts well for the way in which pheromone updates might be incorporated into the collective memory. However, ant algorithms often make use of evaporation as well, and there is no mechanism to account for this in an infinite memory urn model. For this reason, we consider a finite urn. Forcing the contents of the urn to disappear after a configurable amount of time mimics one way in which an ant algorithm could evaporate pheromone.

3.1 Finite Urn Model

Consider an urn that may contain balls of C different colours labelled $0, 1, \ldots, C-1$ in an ordered array of M slots. Each time a draw is made, any balls in slot $M-1$ are removed, and the contents of slot m are transfered to slot $m+1$, for $0 \leq m \leq M-2$. New balls added to the urn are placed in the now empty slot 0. Let $W_i(n, M)$ represent the total weight, in terms of the amount of balls, of colour i in the urn at time n. Moreover, let

$$W(n, M) = (W_0(n, M), W_1(n, M), \ldots, W_{C-1}(n, M))$$

be a vector representing the overall urn composition. Typically we are interested in W at the times immediately following draws of the urn since this is the only time updates occur. Ultimately, the urn process we are interested in is $(W(n, M))_{n=0}^{\infty}$, where n indexes the number of draws, but we do note that it is entirely possible to

view the urn in real or continuous time. Later if we choose to derive expressions in real time we will indicate this clearly by explicitly showing the dependence on continuous time t in place of n. For now, unless otherwise specified we will assume all time dependent quantities are with respect to the discrete time n which counts the number of draws.

The main difference in this slotted urn is in the emphasis placed on the sequence of draws. In our formulation, as with other work, pre-specified rules govern how much weight of each colour is placed in the urn based on the colour drawn. But this implies that the composition of the finite urn can be completely determined at any time simply by knowing the outcomes of the last M draws. Hence, also playing a central role is the outcome of each draw. Let $X_m(n, M)$ represent the ball colour drawn m draws previously. For a finite urn with memory M, at step n it is sufficient to track the values of $X_0(n, M), X_1(n, M), \ldots, X_{M-1}(n, M)$. In fact, for this reason we define the stochastic process representing our urn to be the M-tuple

$$X(n, M) = (X_0(n, M), X_1(n, M), \ldots, X_{M-1}(n, M)).$$

With this view the contents of our ordered array of M slots are no longer the urn contents directly, but rather reflect only the past history of M draws. Thus, slot m, for $m = 0, \ldots, M-1$, contains the colour of the ball drawn m draws previous to the current time. Henceforth, this will be our view of the M slots.

We now focus on the update procedure, a critical part of the description of our urn. From state x, say $(x_0, x_1, \ldots, x_{M-1})$, our process can move to states of the form $\tilde{x}_{(i)} = (i, x_0, \ldots, x_{M-2})$. To make this transition we must choose colour i at the n^{th} draw. We propose that the probability of drawing colour i in this situation is

$$p_{x,\tilde{x}_{(i)}}(M) = \frac{w_i(x)}{w(x)} = \frac{\alpha_i + g_i(x)}{w(x)}. \tag{3.1}$$

Here $w_i(x)$ is the number of balls of colour i in state x, and $w(x) = \sum_{i=0}^{C-1} w_i(x)$ is the total number of balls in the urn in state x. We will insist that $w_i(x) \geq 0$ for all i and x. This ensures that the weight of each colour in the urn is always positive. Moreover, we assume $w(x) > 0$ for every x so that $p_{x,\tilde{x}_{(i)}}(M)$ is always defined. We are further assuming that $w_i(x)$ is the sum of two components. Associated with each colour i is a base number of balls, α_i, and a non-negative function $g_i(x)$ which counts the dynamic number of balls of colour i in the urn in state x. We take $w_i(x) = \alpha_i + g_i(x)$ to be the functional weight of colour i. Since $g_i(x)$ in particular is quite arbitrary, yet critical in the description of any urn of this type, we call our process a functional weight finite urn.

Note that g_i is a function of a state of the urn. That is, we allow the number of balls of each colour present in the urn at a given time to depend on the complete history of the last M draws, including the order. This allows some new modelling flexibility. The effect of drawing colour i may change with time, for example, in this setting. In terms of an ant algorithm application, this opens up the possibility of modelling updates that depend on the last M draws in any way beneficial. This allows us to more explicitly control how feedback information is used beyond the time in which it is initially triggered.

Meanwhile, there is still a sense of updating the number of balls of colour i from one step to the next. Moving from one state, say x, to another state, say $\tilde{x}_{(i)}$, causes a net change of $w_j(\tilde{x}_{(i)}) - w_j(x) = g_j(\tilde{x}_{(i)}) - g_j(x)$ balls of colour j to be either added or removed depending on whether the sign of the difference is positive or negative

respectively.

In this more abstract view we see that the weight functions become critical in determining the evolution of our urn. At any time we can determine the composition of the urn, $W(n, M)$, provided we know the base weights, say $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_{C-1})$, the functional weight components and the current state. With this in mind our process $X(n, M)$ really more accurately represents the feedback in terms of the history of outcomes, not a measure of the amount of each colour ball in the urn. The actual urn composition, we say, can be calculated at any time provided we know the current state of the feedback. Moreover, if we find a stationary distribution for $X(n, M)$, we can easily construct a stationary result for $W(n, M)$ since each component is simply $W_i(n, M) = \alpha_i + g_i(X(n, M))$.

Finally, although we initially described our urn as containing integer numbers of balls for conceptual simplicity, in reality we allow all values reflecting an amount of balls to be real non-negative numbers. This does not present a problem since we have outlined how our process actually tracks outcomes of draws anyway, not actual numbers of balls. Whether the amount of each colour in the urn is real valued or not simply depends on whether the weight functions are real valued or not respectively. At any rate, from here on we will always speak of a weight of colour i in place of a number of balls of colour i, since the former term is more general than the latter.

3.1.1 Null Ball System

For technical reasons, we now describe an alternate strategy for making selections from the urn. We note that urn models are often constrained such that the total

number of balls, or in our context the total weight, added at each step is fixed. This has the effect of simplifying derivations, especially in the case of finite urns. If the net number of balls added at each step is constant then the total number of balls in a finite urn is fixed at all times. We wish to study urns that do not have this restriction imposed. In order to do so, however, we include a device that mimics the same behaviour. In this new scheme, we imagine there is always a null ball present that contributes exactly enough weight to the urn so that the total weight is fixed. The constant value is taken to be the largest weight of any state in a corresponding urn without a null ball. Let this value be $w' = \max_x w(x)$, where $w(x)$ is as before, the weight of state x in an urn without a null ball. The weight of colour i in the urn remains unchanged, but the probability of choosing colour i becomes

$$p'_{x,\tilde{x}_{(i)}}(M) = \frac{w_i(x)}{w'} = \frac{\alpha_i + g_i(x)}{w'}, \tag{3.2}$$

after accounting for the additional weight contributed by the null ball. At the same time, if the null ball is selected at any draw, with probability

$$p'_{x,x}(M) = \frac{w' - w(x)}{w'}, \tag{3.3}$$

we hold in the current state.

We will distinguish between the original selection mechanism and this new scheme by calling the former, without a null ball, the unprimed system, and the latter, with a null ball, the primed system. Notation will always reflect this convention. Quantities referring to an unprimed urn will always be unprimed while those referring to a primed urn will always be primed. Most importantly, for instance, the primed functional

weight finite urn is

$$X'(n, M) = (X'_0(n, M), X'_1(n, M), \ldots, X'_{C-1}(n, M)).$$

Moreover, the concentration of weights for each colour type in a null ball urn process will be denoted by

$$W'(n, M) = (W'_0(n, M), W'_1(n, M), \ldots, W'_{C-1}(n, M)).$$

3.1.2 Special Cases

Above we have described both an unprimed and primed system. The purpose of the primed system is to handle cases where the weight of each state varies. It can happen that $w(x)$ is already constant for all x. We will often find that results greatly simplify in this constant weight special case.

Next, we may consider cases in which we take only specific information from the state we have constructed. So far we have described an urn in which the functional weight of each colour can depend on the state in any way. The state, we recall, consists of a sequence of the outcomes of the last M draws. It includes information about what colours were drawn and in what order. Sometimes the order is not important, rather, it is the numbers of each colour that are critical. For these situations we define

$$s(x) = (s_0(x), s_1(x), \ldots, s_{C-1}(x))$$

to be a vector that counts the number of slots containing each colour in state x. In this case there are many states, each with the same total numbers of slots of each colour

but in different permutations, that correspond to the same vector, $s(x)$. Specifically, there are $\binom{M}{s_0,\ldots,s_{C-1}}$ states x with $s = (s_0, s_1, \ldots, s_{C-1})$. For each vector s, it is useful to declare one state, call it $x^*(s)$, to be the representative of the class. We assume $x^*(s)$ is such that the first s_0 components are colour 0, the next s_1 components are colour 1, etc, until finally the last s_{C-1} components are colour $C-1$.

It is more difficult to model the evolution of the urn when the weight of colour i may depend on a state in an arbitrary manner. Dependence on state through the number of slots of each colour is more tractable. We will focus the majority of attention on this case in which the order is not important. We will term this class of urns exchangeable, owing to the fact that permuting the slots of a given state by definition cannot change the value of the functional weight component in these cases. That is, we assume $g_i(x)$ is actually a function of $s(x)$ in this class.

Furthermore, the functional weight $g_i(x)$ might only depend on the vector $s(x)$ through the single component $s_i(x)$. That is, the weight $w_i(x)$ may be determined completely once $s_i(x)$ is known. In this case, which we will call the definite form class, we will see later that more can be said about the steady state of our urn.

We also observe that dependence only through s does not guarantee that drawing a given colour always adds the same weight of each colour, a property of traditional urns which are often characterized by a ball addition matrix that supplies this information [36]. For this we require another provision in our formulation. If we imagine cases in which the functional weight $g_i(x)$ is a linear function of the components in s, then it is equivalent to a situation in which the same weight of each colour is added each time a given colour is drawn. In this special linear case the coefficients of the functional weight represent the individual weight added. To see this, suppose we have

$$g_i(x) = \Delta_{i0}s_0(x) + \Delta_{i1}s_1(x) + \cdots + \Delta_{i,C-1}s_{C-1}(x).$$

Then we can think of Δ_{ij} as the weight of colour i added given that colour j is drawn. A weight matrix created from the coefficients in the form

$$\Delta = \begin{bmatrix} \Delta_{00} & \Delta_{01} & \cdots & \Delta_{0,C-1} \\ \Delta_{10} & \Delta_{11} & \cdots & \Delta_{1,C-1} \\ \vdots & \vdots & \ddots & \vdots \\ \Delta_{C-1,0} & \Delta_{C-1,1} & \cdots & \Delta_{C-1,C-1} \end{bmatrix} \tag{3.4}$$

plays the analogous role to the traditional ball addition matrix in the existing literature on urns.

To keep these various cases straight, we now summarize the possibilities in diagram form. Figure 3.1 depicts a break down of the set $\mathcal{G}$, which represents all general weight functions. In the diagram, $\mathcal{C}$ is the subset of functions with constant weight $w(x)$. $\mathcal{E}$ represents those urns where the weight is a function of the number of slots of each colour only and the order associated with the state is not considered. Put another way, $\mathcal{E}$ is the set where the random variable components of $X(M)$ are exchangeable. Next some subsets within the exchangeable class are highlighted. Set $\mathcal{D}$, the definite forms set, has a further restriction whereby the functional weight of colour i can be determined completely by $s_i(x)$ alone. Finally, $\mathcal{L}$ is the subset in which the functional weight component $g_i(x) = \sum_{j=0}^{C-1} \Delta_{ij}s_j(x)$ is linear.

Throughout most of the remainder of the document we will refer to the classification above. In the next section for example, we see that the set $\mathcal{D}$ plays an important role in

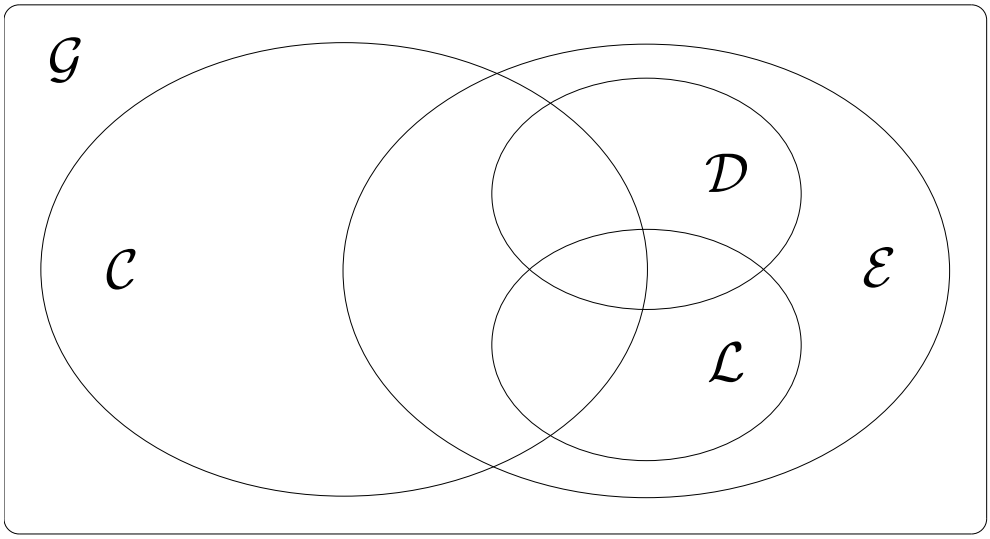

Figure 3.1: Functional weight finite urn types. $\mathcal{G}$ = general forms; $\mathcal{C}$ = constant weight forms; $\mathcal{E}$ = exchangeable forms; $\mathcal{D}$ = definite exchangeable forms; $\mathcal{L}$ = linear exchangeable forms.

determining when we can state the form of the stationary distribution of our process. Similarly, we will spend considerable time exploring urns belonging to the set $\mathcal{L}$ in both the rest of this chapter and continuing in Chapter 4, since the increased structure allows us to develop many quantities of interest. Moreover, various intersections of the sets $\mathcal{C}$, $\mathcal{D}$ and $\mathcal{L}$ lead to simplified expressions. Even in Chapter 5 and Chapter 6 where we generalize the urn beyond what we have seen so far, we may still find it useful to think of special cases of the later modified urn as having their roots in the basic special cases depicted here.

3.2 Stationary Distribution

Over the finite state space of size C^M above, both unprimed and primed methods of evolution describe finite, irreducible Markovian processes, and hence have stationary distributions. We let

$$\pi_{X,x}(M) = \Pr\{X(M) = x\}$$

and

$$\pi_{X',x}(M) = \Pr\{X'(M) = x\}$$

be the steady state distributions of $X(n, M)$ and $X'(n, M)$ respectively. It turns out that there is a simple relationship between $\pi_{X,x}(M)$ and $\pi_{X',x}(M)$ that is explained by considering the proportion of time that the null colour is not selected in the primed system. This probability of not choosing the null ball from state x is $\frac{w(x)}{w'}$. Using this simple observation we have

$$\pi_{X,x}(M) = \frac{\frac{w(x)\pi_{X',x}(M)}{w'}}{\sum_{\tilde{x}} \frac{w(\tilde{x})\pi'_{X',\tilde{x}}(M)}{w'}}.$$

If we define $p'_{null}(M)$ to be the probability of drawing the null ball in a primed finite urn with memory M, we can even say that

$$\sum_{\tilde{x}} \frac{w(\tilde{x})\pi_{X',\tilde{x}}(M)}{w'} = (1 - p'_{null}(M)), \tag{3.5}$$

and hence

$$\pi_{X,x}(M) = \frac{1}{(1 - p'_{null}(M))} \frac{w(x)}{w'} \pi_{X',x}(M). \tag{3.6}$$

Another way of thinking of Equation 3.6 is that it says $\frac{\pi_{X,x}(M)}{w(x)}$ and $\frac{\pi_{X',x}(M)}{w'}$ are proportional to one another. With this in mind, we will use

$$\pi_{X,x}(M) = G(M)^{-1} w(x) \gamma(x)$$

and

$$\pi_{X',x}(M) = G'(M)^{-1} w' \gamma(x)$$

for some constants $G(M)^{-1}$ and $G'(M)^{-1}$ and function $\gamma(x)$.

Next we confirm that

$$\frac{\pi_{X',x}(M)}{\pi_{X,x}(M)} = \frac{G'(M)^{-1} w' \gamma(x)}{G(M)^{-1} w(x) \gamma(x)} = \frac{G'(M)^{-1} w'}{G(M)^{-1} w(x)} = \frac{w'(1 - p'_{null}(M))}{w(x)},$$

so we may also use the relationship

$$\frac{G'(M)^{-1}}{G(M)^{-1}} = (1 - p'_{null}(M)) \tag{3.7}$$

later as necessary.

We now turn our attention to the task of finding $\pi_X(M)$ and $\pi_{X'}(M)$. We have already outlined all possible transitions in either the unprimed or primed urns. Consider now building the single step transition matrices from these in either case. In the unprimed case we use $p_{x,\tilde{x}}(M)$ as defined in Equation 3.1 to construct

$$P(M) = \left(p_{x,\tilde{x}}(M)\right)_{x,\tilde{x}}, \tag{3.8}$$

while in the primed case we use $p'_{x,\tilde{x}}(M)$ as described in the two cases of Equation 3.2 and Equation 3.3 to construct

$$P'(M) = \left(p'_{x,\tilde{x}}(M)\right)_{x,\tilde{x}}. \tag{3.9}$$

The matrices $P(M)$ and $P'(M)$ are sufficient for finding the stationary distributions $\pi_X(M)$ and $\pi_{X'}(M)$ using standard theory of Markov chains. In general, numerical methods may be required in the calculation of solutions.

In the definite exchangeable special case, however, we can explicitly state the algebraic form of the stationary probabilities $\pi_{X,x}(M)$ and $\pi_{X',x}(M)$ exactly. This class is of a reasonable size since it includes at least all urns also in the intersection of it with the linear exchangeable class. Recall, members of the linear set have a corresponding weight matrix. Suppose that in the class $\mathcal{L} \cap \mathcal{D}$ the weight matrix is of the form

$$\Delta = \begin{bmatrix} \Delta_{00} & \delta_0 & \cdots & \delta_0 \\ \delta_1 & \Delta_{11} & \cdots & \delta_1 \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{C-1} & \delta_{C-1} & \cdots & \Delta_{C-1,C-1} \end{bmatrix}. \tag{3.10}$$

Note that we have

$$g_i(x) = s_i(x)\Delta_{ii} + (M - s_i(x))\delta_i,$$

and the number of slots of a given colour completely determines the weight contributed. This last fact is true for the larger set $\mathcal{D}$ as well. That is, the number of

slots of each colour type in any definite exchangeable class becomes critical. For this reason, we define $S_i(n, M)$ and $S'_i(n, M)$ to be the number of slots containing colour i at time n in unprimed and primed urns with memory M respectively. Moreover, we let $s_i(x)$ count the number of slots of colour i in state x. Then the functional weight of colour i can be written as

$$g_i(x) = h_i(s_i(x))$$

for some function $h_i(s_i)$.

These additional definitions allow us to view both the unprimed and primed urns over the state space of vectors $x = (x_0, x_1, \ldots, x_{M-1})$, in which order is important, or over a smaller state space of vectors of the form $s = (s_0, s_1, \ldots, s_{C-1})$. Over this alternative state space our finite urn process is again Markovian, and there exists corresponding stationary distributions which we will call

$$\pi_{S,s}(M) = \Pr\{S(M) = s\}$$

and

$$\pi_{S',s}(M) = \Pr\{S'(M) = s\}$$

in the unprimed and primed versions respectively. Recalling our definition of exchangeability, it is easy to see that we must have

$$\binom{M}{s_0, \ldots, s_{C-1}} \pi_{X,x}(M) = \pi_{S,s}(M) \tag{3.11}$$

and

$$\binom{M}{s_0,\ldots,s_{C-1}}\pi_{X',x}(M) = \pi_{S',s}(M). \tag{3.12}$$

Specifically, there are exactly $\binom{M}{s_0,\ldots,s_{C-1}}$ states in the form x for each corresponding state in the form s, each with the same probability.

Now, we turn our attention to finding the stationary distribution using global balance. By considering all transitions from states of the form $\tilde{x}^{(i)} = (x_1, x_2, \ldots, x_{M-1}, i)$ to the state $x = (x_0, x_1, \ldots, x_{M-1})$, in the following lemmas we effectively show that we can take the component $\gamma(x)$ above to be given by

$$\gamma(x) = \prod_{i=0}^{C-1}\left(\prod_{l_i=0}^{s_i(x)-1}(\alpha_i + h_i(l_i))\right),$$

where $h_i(s_i)$ is defined as above.

Lemma 3.1 *In the class $\mathcal{D}$ we have*

$$\pi_{X,x}(M) = G(M)^{-1}\left(\sum_{k=0}^{C-1}(\alpha_k + h_k(s_k(x)))\right)\prod_{i=0}^{C-1}\left(\prod_{l_i=0}^{s_i(x)-1}(\alpha_i + h_i(l_i))\right)$$

with

$$G(M) = \sum_{\tilde{x}}\left(\sum_{k=0}^{C-1}(\alpha_k + h_k(s_k(\tilde{x})))\right)\prod_{i=0}^{C-1}\left(\prod_{l_i=0}^{s_i(\tilde{x})-1}(\alpha_i + h_i(l_i))\right),$$

and

$$\pi_{S,s}(M) = G(M)^{-1}\binom{M}{s_0,\ldots,s_{C-1}}\left(\sum_{k=0}^{C-1}(\alpha_k + h_k(s_k))\right)\prod_{i=0}^{C-1}\left(\prod_{l_i=0}^{s_i-1}(\alpha_i + h_i(l_i))\right)$$

with

$$G(M) = \sum_{\tilde{s}}\binom{M}{\tilde{s}_0,\ldots,\tilde{s}_{C-1}}\left(\sum_{k=0}^{C-1}(\alpha_k + h_k(\tilde{s}_k))\right)\prod_{i=0}^{C-1}\left(\prod_{l_i=0}^{\tilde{s}_i-1}(\alpha_i + h_i(l_i))\right).$$

Proof *For the first statement we just need to verify the global balance expression for the unprimed system. That is, we need to show that*

$$\begin{aligned}\pi_{X,x}(M) &= p_{\tilde{x}^{(0)},x}(M)\pi_{X,\tilde{x}^{(0)}}(M) + p_{\tilde{x}^{(1)},x}(M)\pi_{X,\tilde{x}^{(1)}}(M) \\ &\quad + \ldots + p_{\tilde{x}^{(C-1)},x}(M)\pi_{X,\tilde{x}^{(C-1)}}(M).\end{aligned}$$

But since for $i \neq x_0$, state $\tilde{x}^{(i)}$ has one more component of colour i and one less component of colour x_0 compared with state x, we can write

$$\begin{aligned}\pi_{X,x}(M) &= p_{\tilde{x}^{(x_0)},x}(M)\pi_{X,x}(M) + \sum_{i\neq x_0} p_{\tilde{x}^{(i)},x}(M)\pi_{X,\tilde{x}^{(i)}}(M) \\ &= p_{\tilde{x}^{(x_0)},x}(M)\pi_{X,x}(M) + \sum_{i\neq x_0} p_{\tilde{x}^{(i)},x}(M)\pi_{X,x}(M)\frac{w_i(x)}{w_{x_0}(\tilde{x}^{(i)})}\frac{w(\tilde{x}^{(i)})}{w(x)}.\end{aligned}$$

Which is equivalent to

$$\begin{aligned}1 &= p_{\tilde{x}^{(x_0)},x}(M) + \sum_{i\neq x_0} p_{\tilde{x}^{(i)},x}(M)\frac{w_i(x)}{w_{x_0}(\tilde{x}^{(i)})}\frac{w(\tilde{x}^{(i)})}{w(x)} \\ &= \frac{w_{x_0}(\tilde{x}^{(x_0)})}{w(\tilde{x}^{(x_0)})} + \sum_{i\neq x_0}\frac{w_{x_0}(\tilde{x}^{(i)})}{w(\tilde{x}^{(i)})}\frac{w_i(x)}{w_{x_0}(\tilde{x}^{(i)})}\frac{w(\tilde{x}^{(i)})}{w(x)} \\ &= \frac{w_{x_0}(x)}{w(x)} + \sum_{i\neq x_0}\frac{w_i(x)}{w(x)} \\ &= \frac{w(x)}{w(x)}.\end{aligned}$$

The second statement follows from the first and Equation 3.11.

□

Lemma 3.2 *In the class $\mathcal{D}$ we have*

$$\pi_{X',x}(M) = G'(M)^{-1} w' \prod_{i=0}^{C-1} \left(\prod_{l_i=0}^{s_i(x)-1} (\alpha_i + h_i(l_i)) \right)$$

with

$$G'(M) = \sum_{\tilde{x}} w' \prod_{i=0}^{C-1} \left(\prod_{l_i=0}^{s_i(\tilde{x})-1} (\alpha_i + h_i(l_i)) \right),$$

and

$$\pi_{S',s}(M) = G'(M)^{-1} \binom{M}{s_0, \ldots, s_{C-1}} w' \prod_{i=0}^{C-1} \left(\prod_{l_i=0}^{s_i-1} (\alpha_i + h_i(l_i)) \right)$$

with

$$G'(M) = \sum_{\tilde{s}} \binom{M}{\tilde{s}_0, \ldots, \tilde{s}_{C-1}} w' \prod_{i=0}^{C-1} \left(\prod_{l_i=0}^{\tilde{s}_i-1} (\alpha_i + h_i(l_i)) \right).$$

Proof *For the first statement we just need to verify the global balance expression for the primed system. That is we need to show that*

$$\begin{aligned} \pi_{X',x}(M) &= p'_{x,x}(M)\pi_{X',x}(M) + p'_{\tilde{x}^{(0)},x}(M)\pi_{X',\tilde{x}^{(0)}}(M) + p'_{\tilde{x}^{(1)},x}(M)\pi_{X',\tilde{x}^{(1)}}(M) \\ &\quad + \ldots + p'_{\tilde{x}^{(C-1)},x}(M)\pi_{X',\tilde{x}^{(C-1)}}(M). \end{aligned}$$

But since for $i \neq x_0$, state $\tilde{x}^{(i)}$ has one more component of colour i and one less component of colour x_0 compared with state x, we can write

$$\begin{aligned} \pi_{X',x}(M) &= p'_{x,x}(M)\pi_{X',x}(M) + p'_{\tilde{x}^{(x_0)},x}(M)\pi_{X',x}(M) + \sum_{i \neq x_0} p'_{\tilde{x}^{(i)},x}(M)\pi_{X',\tilde{x}^{(i)}}(M) \\ &= p'_{x,x}(M)\pi_{X',x}(M) + p'_{\tilde{x}^{(x_0)},x}(M)\pi_{X',x}(M) \\ &\quad + \sum_{i \neq x_0} p'_{\tilde{x}^{(i)},x}(M)\pi_{X',x}(M)\frac{w_i(x)}{w_{x_0}(\tilde{x}^{(i)})}. \end{aligned}$$

Which is equivalent to

$$\begin{aligned} 1 &= p'_{x,x}(M) + p'_{\tilde{x}^{(x_0)},x}(M) + \sum_{i \neq x_0} p'_{\tilde{x}^{(i)},x}(M) \frac{w_i(x)}{w_{x_0}(\tilde{x}^{(i)})} \\ &= \frac{w' - w(x)}{w'} + \frac{w_{x_0}(\tilde{x}^{(x_0)})}{w'} + \sum_{i \neq x_0} \frac{w_{x_0}(\tilde{x}^{(i)})}{w'} \frac{w_i(x)}{w_{x_0}(\tilde{x}^{(i)})} \\ &= \frac{w' - w(x)}{w'} + \frac{w_{x_0}(x)}{w'} + \sum_{i \neq x_0} \frac{w_i(x)}{w'} \\ &= \frac{w'}{w'}. \end{aligned}$$

The second statement follows from the first and Equation 3.12.

□

Note that for urns in the class $\mathcal{L} \cap \mathcal{D}$ we have

$$\begin{aligned} w(x) &= \sum_{k=0}^{C-1} (\alpha_k + h_k(s_k(x))) \\ &= \sum_{k=0}^{C-1} (\alpha_k + s_k(x)\Delta_{kk} + (M - s_k(x))\delta_k) \\ &= \sum_{k=0}^{C-1} (\alpha_k + s_k(x)\Delta_{CS,k}) \end{aligned}$$

and, for urns in the class $\mathcal{L}$,

$$w' = \sum_{k=0}^{C-1} \alpha_k + M\Delta_{MCS}, \tag{3.13}$$

where $\Delta_{CS,k}$ is the sum of the entries in column k of the weight matrix, and $\Delta_{MCS} = \max_k \Delta_{CS,k}$ is the maximum column sum.

Moreover, if the urn is also a constant weight form type, that is, in class $\mathcal{L} \cap \mathcal{D} \cap \mathcal{C}$, we have

$$w(x) = \sum_{i=0}^{C-1} \alpha_i + M\Delta_{CCS} = w',$$

where Δ_{CCS} is the constant column sum of the weight matrix. By Equation 3.10, the constant column sum must also satisfy

$$\Delta_{CCS} = \Delta_{jj} - \delta_j + \sum_{k=0}^{C-1} \delta_k$$

for any column j. But then $\tau_j = \Delta_{jj} - \delta_j$, the on-diagonal, off-diagonal difference, must be constant as well. That is, if we call this constant value τ, we have

$$\tau_j = \Delta_{jj} - \delta_j = \Delta_{CCS} - \sum_{k=0}^{C-1} \delta_k = \tau. \tag{3.14}$$

Furthermore, when an urn belongs to the set $\mathcal{L} \cap \mathcal{D} \cap \mathcal{C}$, we can even find $G(M)$ in closed form. This allows us to simplify the stationary distribution expressions further. We the above observations in mind, note that $\pi_{X,x}(M) = \pi_{X',x}(M)$, $\pi_{S,s}(M) = \pi_{S',s}(M)$ and $G(M) = G'(M)$ in this case, and consider the following result.

Lemma 3.3 *In the class $\mathcal{L} \cap \mathcal{D} \cap \mathcal{C}$ we have*

$$\pi_{X,x}(M) = \pi_{X',x}(M) = \frac{\prod_{i=0}^{C-1} \left(\prod_{l_i=0}^{s_i(x)-1} (\alpha_i + h_i(l_i)) \right)}{\prod_{l=0}^{M-1} \left(\sum_{i=0}^{C-1} (\alpha_i + M\delta_i) + \tau l \right)}$$

and

$$\pi_{S,s}(M) = \pi_{S',s}(M) = \binom{M}{s_0, \ldots, s_{C-1}} \frac{\prod_{i=0}^{C-1} \left(\prod_{l_i=0}^{s_i(x)-1} (\alpha_i + h_i(l_i)) \right)}{\prod_{l=0}^{M-1} \left(\sum_{i=0}^{C-1} (\alpha_i + M\delta_i) + \tau l \right)}.$$

Proof *For the first statement we explore the normalization constant as given in Lemmas 3.1 and 3.2. We fix M for the moment and consider the first M draws of a special urn where the base weight of each colour i is $\alpha_i + M\delta_i$, and the weight matrix is τI. The probability of drawing exactly s_i balls of colour i, in some order so that the state after M draws is $s = (s_0, \ldots, s_{C-1})$, for this special urn is simply*

$$\binom{M}{s_0, \ldots, s_{C-1}} \frac{\prod_{i=0}^{C-1} \left(\prod_{l_i=0}^{s_i-1} (\alpha_i + M\delta_i + \tau l_i) \right)}{\prod_{l=0}^{M-1} \left(\sum_{i=0}^{C-1} (\alpha_i + M\delta_i) + \tau l \right)}.$$

But this means we have

$$\sum_s \binom{M}{s_0, \ldots, s_{C-1}} \frac{\prod_{i=0}^{C-1} \left(\prod_{l_i=0}^{s_i-1} (\alpha_i + M\delta_i + \tau l_i) \right)}{\prod_{l=0}^{M-1} \left(\sum_{i=0}^{C-1} (\alpha_i + M\delta_i) + \tau l \right)} = 1,$$

which implies that

$$\sum_s \binom{M}{s_0, \ldots, s_{C-1}} \prod_{i=0}^{C-1} \left(\prod_{l_i=0}^{s_i-1} (\alpha_i + M\delta_i + \tau l_i) \right) = \prod_{l=0}^{M-1} \left(\sum_{i=0}^{C-1} (\alpha_i + M\delta_i) + \tau l \right)$$

for every M. Hence, for an urn in the class $\mathcal{L} \cap \mathcal{D} \cap \mathcal{C}$ we can write

$$\begin{aligned} G(M) &= \sum_s \binom{M}{s_0, \ldots, s_{C-1}} \left(\sum_{i=0}^{C-1} \alpha_i + M\Delta_{CCS} \right) \prod_{i=0}^{C-1} \left(\prod_{l_i=0}^{s_i-1} (\alpha_i + M\delta_i + \tau l_i) \right) \\ &= \left(\sum_{i=0}^{C-1} \alpha_i + M\Delta_{CCS} \right) \sum_s \prod_{i=0}^{C-1} \binom{M}{s_0, \ldots, s_{C-1}} \left(\prod_{l_i=0}^{s_i-1} (\alpha_i + M\delta_i + \tau l_i) \right) \\ &= \left(\sum_{i=0}^{C-1} \alpha_i + M\Delta_{CCS} \right) \prod_{l=0}^{M-1} \left(\sum_{i=0}^{C-1} (\alpha_i + M\delta_i) + \tau l \right). \end{aligned}$$

The second statement follows from the first and Equations 3.11 and 3.12.

□

3.3 First and Second Order Slot Probabilities

For this section we will restrict our view to functional weight finite urns of the linear exchangeable type. Recall that within this set each urn is characterized by a weight matrix, Δ. In this section we are interested in $S_i(M)$ and $S_i'(M)$. Specifically, we will derive expressions for the first order quantity $E[S_i(M)]$, $0 \leq i \leq C-1$ and the second order quantity $E[S_i(M)S_j(M)]$, $0 \leq i, j \leq C-1$.

3.3.1 First Order Calculations

Later we will look at a definite exchangeable urn, but for now, suppose we only know it is exchangeable. In steady state, the expected number of slots containing colour i is simply M times the probability that a slot contains colour i. We will see this more directly in the following. We begin by defining $q_i(M)$ and $q_i'(M)$ to be the probability of slot 0 containing colour i in an unprimed and primed urn respectively. Next, we calculate:

$$\begin{aligned}
E[S_i(M)] &= \sum_x s_i(x)\pi_{X,x}(M) \\
&= \sum_x s_i(x)G(M)^{-1}w(x)\gamma(x) \\
&= \sum_s s_i G(M)^{-1}\binom{M}{s_0,\ldots,s_{C-1}}w(s)\gamma(s) \\
&= M\sum_{s:s_i\geq 1} G(M)^{-1}\binom{M-1}{s_0,\ldots,s_i-1,\ldots,s_{C-1}}w(s)\gamma(s) \\
&= M\sum_{x:x_0=i} G(M)^{-1}w(x)\gamma(x) \\
&= M\sum_{x:x_0=i}\pi_{X,x}(M),
\end{aligned}$$

where x_0 is the contents of slot 0 as usual, and we have abused notation slightly by writing $w(s)$ and $\gamma(s)$, since in the exchangeable case $w(x)$ and $\gamma(x)$ are functions of x only through $s(x)$. So indeed we have

$$E[S_i(M)] = Mq_i(M), \tag{3.15}$$

and by similar calculations,

$$E[S'_i(M)] = Mq'_i(M). \tag{3.16}$$

Note that although we have highlighted slot 0 here for concreteness, any single slot will suffice. In fact, for this reason we can think of $q_i(M)$ and $q'_i(M)$ referring to the probability that any single slot contains colour i in an unprimed and primed urn.

Next, we define $p_i(M)$ and $p'_i(M)$ to be the probability of drawing colour i in an unprimed and primed urn respectively. We note that although $p_i(M) = q_i(M)$, $p'_i(M) \neq q'_i(M)$ in general because of the presence of the null ball.

We can now relate the various quantities defined so far by considering them in terms of the transitions that occur in a null ball system in steady state. We have

$$p'_i(M) = (1 - p'_{null}(M))q_i(M), \tag{3.17}$$

for example, since $p'_i(M)$ is the proportion of transitions that lead to a state with colour i in the first slot, excluding the times when the null ball is chosen. Furthermore, by calculating $q'_i(M)$ by conditioning on the ball colour drawn, we obtain:

$$q'_i(M) = p'_i(M) + p'_{null|i}(M)q'_i(M),$$

or

$$q'_i(M) = \frac{1}{(1 - p'_{null|i}(M))} p'_i(M), \tag{3.18}$$

where $p'_{null|i}(M)$ is the probability of drawing the null ball given slot 0 contains colour i.

Combining Equations 3.17 and 3.18 we have

$$q'_i(M) = \frac{(1 - p'_{null}(M))}{(1 - p'_{null|i}(M))} q_i(M) = r'_i(M) q_i(M), \tag{3.19}$$

where we use $r'_i(M) = \frac{(1-p'_{null}(M))}{(1-p'_{null|i}(M))}$.

Let us now calculate $q_i(M)$, assuming further that our urn is in the linear class, as

$$\begin{aligned}
q_i(M) &= \sum_x \frac{w_i(x)}{w(x)} \pi_{X,x}(M) \\
&= \sum_x \frac{w_i(x)}{w(x)} G(M)^{-1} w(x) \gamma(x) \\
&= \frac{G(M)^{-1}}{w' G'(M)^{-1}} \sum_x w_i(x) G'(M)^{-1} w' \gamma(x) \\
&= \frac{G(M)^{-1}}{w' G'(M)^{-1}} \sum_x \left(\alpha_i + \sum_{j=0}^{C-1} \Delta_{ij} s_j(x) \right) \pi_{X',x}(M) \\
&= \frac{G(M)^{-1}}{w' G'(M)^{-1}} \left(\alpha_i + \sum_{j=0}^{C-1} \Delta_{ij} E[S'_j(M)] \right) \\
&= \frac{G(M)^{-1}}{w' G'(M)^{-1}} \left(\alpha_i + \sum_{j=0}^{C-1} \Delta_{ij} M q'_j(M) \right).
\end{aligned}$$

Recalling the relation $w' = \left(\sum_{k=0}^{C-1} \alpha_k \right) + M \Delta_{MCS}$ and Equations 3.7 and 3.19, we make substitutions and simplify to

$$\left(\left(\sum_{k=0}^{C-1}\alpha_k\right)+M\Delta_{MCS}\right)(1-p'_{null}(M))q_i(M) = \alpha_i+\sum_{j=0}^{C-1}\Delta_{ij}Mq'_j(M)$$
$$= \alpha_i+\sum_{j=0}^{C-1}\Delta_{ij}Mr'_j(M)q_j(M).$$

Finally, dividing through by M we obtain

$$\left(\frac{\left(\sum_{k=0}^{C-1}\alpha_k\right)}{M}+\Delta_{MCS}\right)(1-p'_{null}(M))q_i(M)=\frac{\alpha_i}{M}+\sum_{j=0}^{C-1}\Delta_{ij}r'_j(M)q_j(M). \quad (3.20)$$

This expression does not provide a way to solve for $q_i(M)$ itself. However, it is a critical relation for theory presented later.

It is interesting to investigate further refinements of this expression. Reviewing Figure 3.1 we see that, when we take into account all possible intersections, the set $\mathcal{L}$ is partitioned into four subsets. The expression above for $q_i(M)$ holds for arbitrary weight functions in the class $\mathcal{L}$, but in the following, we simplify the result for each of $\mathcal{L}\cap\mathcal{C}$, $\mathcal{L}\cap\mathcal{D}$ and $\mathcal{L}\cap\mathcal{D}\cap\mathcal{C}$.

In the linear form case, requiring each state to have the same weight is equivalent to all columns of Δ having the same sum, Δ_{CCS}. The constant weight in this case is then $w(x)=\left(\sum_{k=0}^{C-1}\alpha_k\right)+M\Delta_{CCS}$. Moreover, the probability of drawing the null ball at any stage is 0. This implies that $p'_{null}(M)=0$ and $p'_{null|i}(M)=0$, for all i. We can also say $G(M)^{-1}=G'(M)^{-1}$ and $q_i(M)=q'_i(M)$. These facts allow us to avoid talking about the primed system entirely in our derivations, greatly simplifying the development of the expression. If we repeat the derivation above we find that

$$\left(\frac{\left(\sum_{k=0}^{C-1}\alpha_k\right)}{M}+\Delta_{CCS}\right)q_i(M)=\frac{\alpha_i}{M}+\sum_{j=0}^{C-1}\Delta_{ij}q_j(M). \quad (3.21)$$

The crucial observation here is that unlike the general expression, this result specifies a set of equations involving unprimed quantities only. Hence, we have a system that can be solved, without further delay, to determine each $q_i(M)$. We note that M equations formed by using each possible i in the left hand side above leads to a linearly dependent system. However, we can always substitute $\sum_{i=0}^{C-1} q_i(M) = 1$ for one of the M equations to form a linearly independent set that can be solved.

Next, in the linear and definite case, that is $\mathcal{L} \cap \mathcal{D}$, we can say even more about the actual form of $q_i(M)$. Recall that in this case Δ takes the form given in Equation 3.10.

This time we have

$$\begin{aligned}
q_i(M) &= \sum_x \frac{w_i(x)}{w(x)} \pi_{X,x}(M) \\
&= \sum_x \frac{w_i(x)}{w(x)} G(M)^{-1} w(x) \gamma(x) \\
&= \frac{G(M)^{-1}}{w' G'(M)^{-1}} \sum_x w_i(x) G'(M)^{-1} w' \gamma(x) \\
&= \frac{G(M)^{-1}}{w' G'(M)^{-1}} \sum_x (\alpha_i + M\delta_i + s_i(x)(\Delta_{ii} - \delta_i)) \pi_{X',x}(M) \\
&= \frac{G(M)^{-1}}{w' G'(M)^{-1}} \left((\alpha_i + M\delta_i) + (\Delta_{ii} - \delta_i) E[S_i'(M)] \right) \\
&= \frac{G(M)^{-1}}{w' G'(M)^{-1}} \left((\alpha_i + M\delta_i) + (\Delta_{ii} - \delta_i) M q_i'(M) \right),
\end{aligned}$$

which leads to

$$q_i(M) \left(\left(\left(\sum_{k=0}^{C-1} \alpha_k \right) + M\Delta_{MCS} \right) (1 - p'_{null}(M)) - (\Delta_{ii} - \delta_i) M r_i'(M) \right) = (\alpha_i + M\delta_i)$$

or

$$q_i(M) = \frac{\left(\frac{\alpha_i}{M} + \delta_i\right)}{\left(\left(\frac{\left(\sum_{k=0}^{C-1} \alpha_k\right)}{M} + \Delta_{MCS}\right)(1 - p'_{null}(M)) - \tau_i r'_i(M)\right)}. \tag{3.22}$$

where we have used $\tau_i = (\Delta_{ii} - \delta_i)$, as previously defined.

Once again we cannot immediately use this to solve for $q_i(M)$ as it contains unknown primed quantities from the introduction of the null ball. Nevertheless, it is interesting because $q_i(M)$ is not presented in terms of all other $q_j(M)$ values.

Finally, if we restrict our view to the set $\mathcal{L} \cap \mathcal{D} \cap \mathcal{C}$, we can do even better. By using Equation 3.14 we find

$$\begin{aligned} q_i(M) &= \frac{\left(\frac{\alpha_i}{M} + \delta_i\right)}{\left(\left(\frac{\left(\sum_{k=0}^{C-1} \alpha_k\right)}{M} + \Delta_{CCS}\right) - \tau_i\right)} \\ &= \frac{\left(\frac{\alpha_i}{M} + \delta_i\right)}{\left(\left(\frac{\left(\sum_{k=0}^{C-1} \alpha_k\right)}{M} + \Delta_{CCS}\right) - \left(\Delta_{CCS} - \sum_{k=0}^{C-1} \delta_k\right)\right)} \\ &= \frac{\left(\frac{\alpha_i}{M} + \delta_i\right)}{\left(\frac{\left(\sum_{k=0}^{C-1} \alpha_k\right)}{M} + \sum_{k=0}^{C-1} \delta_k\right)}. \end{aligned} \tag{3.23}$$

The trade-off for the most restrictive conditions of the functional weight component is that this is the simplest expression so far. We have an explicit expression for $q_i(M)$ in terms of completely known quantities.

3.3.2 Second Order Calculations

Next we want to calculate $E[S_i(M)S_j(M)]$ for all i, j. In analogy with the first order calculations above, calculating our desired expected value will boil down to finding a

specific probability, this time a measure over two slot contents. We define $q_{ij}(M)$ and $q'_{ij}(M)$ to be the probability that slot 0 contains colour i and slot 1 contains colour j in an unprimed and primed urn respectively. We begin calculating as before, but, this time we will need to distinguish the cases $i = j$ and $i \neq j$.

If $i = j$, we have

$$\begin{aligned}
&E[S_i(M)S_i(M)] \\
&\quad = E[S_i(M)(S_i(M)-1)] + E[S_i(M)] \\
&\quad = \sum_x s_i(x)(s_i(x)-1)\pi_{X,x}(M) + Mq_i(M) \\
&\quad = \sum_x s_i(x)(s_i(x)-1)G(M)^{-1}w(x)\gamma(x) + Mq_i(M) \\
&\quad = \sum_s s_i(s_i-1)G(M)^{-1}\binom{M}{s_0,\ldots,s_{C-1}}w(s)\gamma(s) + Mq_i(M) \\
&\quad = M(M-1)\sum_{s:s_i\geq 2} G(M)^{-1}\binom{M-2}{s_0,\ldots,s_i-2,\ldots,s_{C-1}}w(s)\gamma(s) + Mq_i(M) \\
&\quad = M(M-1)\sum_{x:x_0=i,x_1=i} G(M)^{-1}w(x)\gamma(x) + Mq_i(M) \\
&\quad = M(M-1)\sum_{x:x_0=i,x_1=i} \pi_{X,x}(M) + Mq_i(M),
\end{aligned}$$

where we take x_0 to be the contents of slot 0 and x_1 to be the contents of slot 1. This is just $Mq_i(M)$ plus $M(M-1)$ times the probability slot 0 has colour i and slot 1 has colour i, which we have defined $q_{ii}(M)$. Of course we could similarly directly calculate $E[S'_i(M)S'_i(M)] = M(M-1)q'_{ii}(M) + Mq_i(M)$.

If $i \neq j$, we have

$$\begin{aligned}
&E[S_i(M)S_j(M)] \\
&\quad = \sum_x s_i(x)s_j(x)\pi_{X,x}(M) \\
&\quad = \sum_x s_i(x)s_j(x)G(M)^{-1}w(x)\gamma(x)
\end{aligned}$$

$$
\begin{aligned}
&= \sum_s s_i s_j \binom{M}{s_0, \ldots, s_{C-1}} G(M)^{-1} w(s)\gamma(s) \\
&= M(M-1) \sum_{s: s_i \geq 1, s_j \geq 1} \binom{M-2}{s_0, \ldots, s_i - 1, \ldots, s_j - 1, \ldots, s_{C-1}} G(M)^{-1} w(s)\gamma(s) \\
&= M(M-1) \sum_{x: x_0 = i, x_1 = j} G(M)^{-1} w(x)\gamma(x) \\
&= M(M-1) \sum_{x: x_0 = i, x_1 = j} \pi_{X,x}(M),
\end{aligned}
$$

where this is just $M(M-1)$ times the probability slot 0 has colour i and slot 1 has colour j, which using our defined notation is $q_{ij}(M)$. Of course we could similarly directly calculate $E[S'_i(M)S'_j(M)] = M(M-1)q'_{ij}(M)$.

So altogether we see that

$$E[S_i(M)S_j(M)] = M(M-1)q_{ij}(M) + Mq_i(M)I_{\{i=j\}}, \tag{3.24}$$

and

$$E[S'_i(M)S'_j(M)] = M(M-1)q'_{ij}(M) + Mq'_i(M)I_{\{i=j\}}, \tag{3.25}$$

where we use $I_{\{E\}}$ to represent a standard indicator function for the event E. We once again note that although we use slot 0 and slot 1, any particular pair of slots serve the same purpose. As before, we can think of $q_{ij}(M)$ and $q'_{ij}(M)$ as if they refer to two arbitrarily identified slots of the urn. We also note that as with Equation 3.15 and Equation 3.16, both Equation 3.24 and Equation 3.25 are valid for any exchangeable urn regardless of whether it also belongs to the set $\mathcal{L}$.

We now define $p_{ij}(M)$ and $p'_{ij}(M)$ to be the probability of drawing colour i, with colour j currently in slot 0, in an unprimed and primed urn respectively. We note

that $p_{ij}(M) = q_{ij}(M)$ but that $p'_{ij}(M) \neq q'_{ij}(M)$ in general because of the times when the null ball is selected.

Thinking about the proportions of transitions we can write

$$p'_{ij}(M) = (1 - p'_{null}(M))q_{ij}(M), \tag{3.26}$$

using an argument similar to that used in the first order calculations. Moreover, calculating $q'_{ij}(M)$ by conditioning on the ball colour drawn, we obtain

$$q'_{ij}(M) = p'_{ij}(M) + p'_{null|ij}(M)q'_{ij}(M),$$

or

$$q'_{ij}(M) = \frac{1}{(1 - p'_{null|ij}(M))}p'_{ij}(M), \tag{3.27}$$

where we define $p'_{null|ij}(M)$ to be the probability of drawing the null ball given that slot 0 contains colour i and slot 1 contains colour j.

Combining Equations 3.26 and 3.27 we have

$$q'_{ij}(M) = \frac{(1 - p'_{null}(M))}{(1 - p'_{null|ij}(M))}q_{ij}(M) = r'_{ij}(M)q_{ij}(M), \tag{3.28}$$

where this time the second order object $r'_{ij}(M)$ is the ratio of $(1 - p'_{null}(M))$ to $(1 - p'_{null|ij}(M))$.

Now, for an exchangeable and linear urn we calculate $q_{ij}(M)$ thinking about the probability of drawing colour i given that there is already colour j in the first slot of the queue. We have

$$
\begin{aligned}
q_{ij}(M) &= \sum_{x:x_0=j} \frac{w_i(x)}{w(x)} \pi_{X,x}(M) \\
&= \sum_{x:x_0=j} \frac{w_i(x)}{w(x)} G(M)^{-1} w(x)\gamma(x) \\
&= \frac{G(M)^{-1}}{w'G'(M)^{-1}} \sum_{x:x_0=j} w_i(x) G'(M)^{-1} w'\gamma(x) \\
&= \frac{G(M)^{-1}}{w'G'(M)^{-1}} \sum_{s:s_j\geq 1} w_i(s) G'(M)^{-1} \binom{M-1}{s_0,\ldots,s_j-1,\ldots,s_{C-1}} w'\gamma(s) \\
&= \frac{G(M)^{-1}}{Mw'G'(M)^{-1}} \sum_{s} w_i(s) s_j G'(M)^{-1} \binom{M}{s_0,\ldots,s_{C-1}} w'\gamma(s) \\
&= \frac{G(M)^{-1}}{Mw'G'(M)^{-1}} \sum_{x} w_i(x) s_j(x) G'(M)^{-1} w'\gamma(x) \\
&= \frac{G(M)^{-1}}{Mw'G'(M)^{-1}} \sum_{x} \left(\alpha_i + \sum_{k=0}^{C-1} \Delta_{ik} s_k(x)\right) s_j(x) \pi_{X',x}(M) \\
&= \frac{G(M)^{-1}}{Mw'G'(M)^{-1}} \left(\alpha_i E[S'_j(M)] + \sum_{k=0}^{C-1} \Delta_{ik} E[S'_k(M) S'_j(M)]\right) \\
&= \frac{G(M)^{-1}}{w'G'(M)^{-1}} \left(\alpha_i q'_j(M) + (M-1) \sum_{k=0}^{C-1} \Delta_{ik} q'_{kj}(M) + \Delta_{ij} q'_j(M)\right)
\end{aligned}
$$

or

$$
\begin{aligned}
&\left(\left(\sum_{k=0}^{C-1} \alpha_k\right) + M\Delta_{MCS}\right)(1 - p'_{null}(M)) q_{ij}(M) \\
&= \alpha_i q'_j(M) + (M-1) \sum_{k=0}^{C-1} \Delta_{ik} q'_{kj}(M) + \Delta_{ij} q'_j(M) \\
&= \alpha_i r'_j(M) q_j(M) + (M-1) \sum_{k=0}^{C-1} \Delta_{ik} r'_{kj}(M) q_{kj}(M) + \Delta_{ij} r'_j(M) q_j(M),
\end{aligned}
$$

where we have used Equations 3.7, 3.19 and 3.28 and $w' = \left(\sum_{k=0}^{C-1} \alpha_k\right) + M\Delta_{MCS}$. Dividing through by M we obtain

$$\left(\frac{\left(\sum_{k=0}^{C-1}\alpha_k\right)}{M}+\Delta_{MCS}\right)(1-p'_{null}(M))q_{ij}(M)$$
$$= \frac{\alpha_i r'_j(M)q_j(M)}{M}+\frac{(M-1)}{M}\sum_{k=0}^{C-1}\Delta_{ik}r'_{kj}(M)q_{kj}(M)+\frac{\Delta_{ij}r'_j(M)q_j(M)}{M}. \quad (3.29)$$

This general expression is the second order analogue of Equation 3.20. Like its first order counterpart, the above expression does not point directly to a solution for $q_{ij}(M)$, even if each $q_j(M)$ were known. Also like the first order calculations however, some special cases can be solved, as we will see in the following.

The next step is to consider the subsets of $\mathcal{L}$ as we did when calculating the first order moments. In the constant column sum case we know $w(x)=\left(\sum_{k=0}^{C-1}\alpha_k\right)+M\Delta_{CCS}$ for each state x and there is no null ball, which implies that $p'_{null}(M)=0$, $p'_{null|i}(M)=0$ and $p'_{null|ij}(M)=0$ for all i and j. We these facts in mind, the expression for general $\mathcal{L}$ above reduces to

$$\left(\frac{\left(\sum_{k=0}^{C-1}\alpha_k\right)}{M}+\Delta_{CCS}\right)q_{ij}(M)$$
$$= \frac{\alpha_i q_j(M)}{M}+\frac{(M-1)}{M}\sum_{k=0}^{C-1}\Delta_{ik}q_{kj}(M)+\frac{\Delta_{ij}q_j(M)}{M}. \quad (3.30)$$

We have previously outlined a method for solving for the first order $q_j(M)$ values in this case. With this in mind, the above leads to a second system of equations that can be solved for each $q_{ij}(M)$. This time we will need to replace one of the M^2 equations above with $\sum_{i,j=0}^{C-1}q_{ij}(M)=1$ to yield a linearly independent system.

Next, as in the derivation of $q_i(M)$ in the definite linear case, we can say more about the actual form of $q_{ij}(M)$ here.

We have

$$\begin{aligned}
q_{ij}(M) &= \sum_{x:x_0=j} \frac{w_i(x)}{w(x)} \pi_{X,x}(M) \\
&= \sum_{x:x_0=j} \frac{w_i(x)}{w(x)} G(M)^{-1} w(x)\gamma(x) \\
&= \frac{G(M)^{-1}}{w'G'(M)^{-1}} \sum_{x:x_0=j} w_i(x) G'(M)^{-1} w'\gamma(x) \\
&= \frac{G(M)^{-1}}{w'G'(M)^{-1}} \sum_{s:s_j\geq 1} w_i(s) G'(M)^{-1} \binom{M-1}{s_0,\ldots,s_j-1,\ldots,s_{C-1}} w'\gamma'(s) \\
&= \frac{G(M)^{-1}}{Mw'G'(M)^{-1}} \sum_{s} w_i(s) s_j G'(M)^{-1} \binom{M}{s_0,\ldots,s_{C-1}} w'\gamma'(s) \\
&= \frac{G(M)^{-1}}{Mw'G'(M)^{-1}} \sum_{x} w_i(x) s_j(x) G'(M)^{-1} w'\gamma(x) \\
&= \frac{G(M)^{-1}}{Mw'G'(M)^{-1}} \sum_{x} (\alpha_i + M\delta_i + s_i(x)(\Delta_{ii}-\delta_i)) s_j(x) \pi_{X',x}(M) \\
&= \frac{G(M)^{-1}}{Mw'G'(M)^{-1}} \sum_{x} ((\alpha_i + M\delta_i) s_j(x) + \tau_i s_i(x) s_j(x)) \pi_{X',x}(M) \\
&= \frac{G(M)^{-1}}{Mw'G'(M)^{-1}} ((\alpha_i + M\delta_i) E[S'_j(M)] + \tau_i E[S'_i(M) S'_j(M)]) \\
&= \frac{G(M)^{-1}}{Mw'G'(M)^{-1}} ((\alpha_i + M\delta_i) M q'_j(M) + \tau_i (M(M-1) q'_{ij}(M) + M q'_j(M) I_{\{i=j\}})) \\
&= \frac{G(M)^{-1}}{Mw'G'(M)^{-1}} ((\alpha_i + M\delta_i + \tau_i I_{\{i=j\}}) M q'_j(M) + \tau_i M(M-1) q'_{ij}(M)) \\
&= \frac{G(M)^{-1}}{w'G'(M)^{-1}} ((\alpha_i + M\delta_i + \tau_i I_{\{i=j\}}) q'_j(M) + \tau_i (M-1) q'_{ij}(M))
\end{aligned}$$

This time we solve for $q_{ij}(M)$ and get

$$\begin{aligned}
& q_{ij}(M) \left(\left(\left(\sum_{k=0}^{C-1} \alpha_k \right) + M\Delta_{MCS} \right) (1 - p'_{null}(M)) - (M-1)\tau_i r'_{ij}(M) \right) \\
&\quad = \quad (\alpha_i + M\delta_i + \tau_i I_{\{i=j\}}) r'_j(M) q_j(M)
\end{aligned}$$

or

$$q_{ij}(M) = \frac{\left(\frac{\alpha_i}{M} + \delta_i + \frac{\tau_i I_{\{i=j\}}}{M}\right) r'_j(M) q_j(M)}{\left(\left(\frac{\left(\sum_{k=0}^{C-1} \alpha_k\right)}{M} + \Delta_{MCS}\right)(1 - p'_{null}(M)) - \frac{(M-1)}{M}\tau_i r'_{ij}(M)\right)}. \tag{3.31}$$

While we cannot solve this explicitly, the form is once again intriguing in that it does not specify each $q_{ij}(M)$ in terms of all the others. Rather $q_{ij}(M)$ is isolated.

Finally, we consider the most constrained subset, $\mathcal{L} \cap \mathcal{D} \cap \mathcal{C}$. Making use of Equation 3.14 once more, the above expression for $q_{ij}(M)$ in the case $\mathcal{L} \cap \mathcal{D}$ reduces further to

$$\begin{aligned} q_{ij}(M) &= \frac{\left(\frac{\alpha_i}{M} + \delta_i + \frac{\tau_i I_{\{i=j\}}}{M}\right) q_j(M)}{\left(\left(\frac{\sum_{k=0}^{C-1} \alpha_k}{M} + \Delta_{CCS}\right) - \frac{(M-1)}{M}\tau_i\right)} \\ &= \frac{\left(\frac{\alpha_i}{M} + \delta_i + \frac{\left(\Delta_{CCS} - \sum_{k=0}^{C-1} \delta_k\right) I_{\{i=j\}}}{M}\right) q_j(M)}{\left(\left(\frac{\left(\sum_{k=0}^{C-1} \alpha_k\right)}{M} + \Delta_{CCS}\right) - \frac{(M-1)}{M}\left(\Delta_{CCS} - \sum_{k=0}^{C-1} \delta_k\right)\right)} \\ &= \frac{\left(\frac{\alpha_i}{M} + \delta_i + \frac{\left(\Delta_{CCS} - \sum_{k=0}^{C-1} \delta_k\right) I_{\{i=j\}}}{M}\right) q_j(M)}{\left(\frac{\left(\sum_{k=0}^{C-1} \alpha_k\right)}{M} + \frac{\Delta_{CCS}}{M} + \frac{(M-1)}{M}\sum_{k=0}^{C-1} \delta_k\right)}. \end{aligned} \tag{3.32}$$

Here, as in the analogous first order theory, we have a direct way to calculate $q_{ij}(M)$. Of course we note that it is a simple matter to first determine $q_j(M)$ in this case.

3.4 Mean and Covariance Measures

In this section we investigate some basic measures of our finite urn with memory M. We are in a position, given the above first and second order slot probability

expressions, to contemplate the mean and covariance of both the number of slots and the weight of each colour.

We recall that

$$E[S_i(M)] = Mq_i(M) \tag{3.33}$$

and

$$E[S_i'(M)] = Mq_i'(M). \tag{3.34}$$

Since we have already explored $q_i(M)$ and $q_i'(M)$ in several cases, there is nothing further to do here for the mean of $S_i(M)$ or $S_i'(M)$.

Furthermore, since $W_i(M) = \sum_{j=0}^{C-1} \Delta_{ij} S_j(M)$ it is straightforward to calculate

$$\begin{aligned} E[W_i(M)] &= E[\sum_{j=0}^{C-1} \Delta_{ij} S_j(M)] \\ &= \sum_{j=0}^{C-1} \Delta_{ij} M q_j(M) \\ &= M \left(\sum_{j=0}^{C-1} \Delta_{ij} q_j(M) \right) \end{aligned} \tag{3.35}$$

if we wish. Similarly, we have $W_i'(M) = \sum_{j=0}^{C-1} \Delta_{ij} S_j'(M)$, and hence

$$E[W_i'(M)] = M \left(\sum_{j=0}^{C-1} \Delta_{ij} q_j'(M) \right). \tag{3.36}$$

We also note that if we are in the set $\mathcal{L} \cap \mathcal{D}$ we can write

$$\begin{aligned} E[W_i(M)] &= E[\Delta_{ii} S_i(M) + \delta_i(M - S_i(M))] \\ &= (\Delta_{ii} - \delta_i) E[S_i(M)] + M\delta_i \\ &= \tau_i M q_i(M) + M\delta_i \\ &= M\left(\tau_i q_i(M) + \delta_i\right) \end{aligned} \tag{3.37}$$

and

$$E[W_i'(M)] = M\left(\tau_i q_i'(M) + \delta_i\right). \tag{3.38}$$

for the unprimed and primed cases respectively.

Next, for future reference, we write down covariance expressions. We have

$$\begin{aligned} &E[S_i(M)S_j(M)] - E[S_i(M)]E[S_j(M)] \\ &= M(M-1)q_{ij}(M) + Mq_i(M)I_{\{i=j\}} - Mq_i(M)Mq_j(M) \\ &= M\Big((M-1)q_{ij}(M) + q_i(M)I_{\{i=j\}} - Mq_i(M)q_j(M)\Big) \end{aligned} \tag{3.39}$$

and, similarly

$$\begin{aligned} &E[S_i'(M)S_j'(M)] - E[S_i'(M)]E[S_j'(M)] \\ &= M\Big((M-1)q_{ij}'(M) + q_i'(M)I_{\{i=j\}} - Mq_i'(M)q_j'(M)\Big). \end{aligned} \tag{3.40}$$

Next we calculate the general expression for the covariance of $W_i(M)$ and $W_j(M)$ as

$$
\begin{aligned}
&E[W_i(M)W_j(M)] - E[W_i(M)]E[W_j(M)] \\
&= E\left[\left(\sum_{k=0}^{C-1}\Delta_{ik}S_k(M)\right)\left(\sum_{l=0}^{C-1}\Delta_{il}S_l(M)\right)\right] \\
&\quad - E\left[\left(\sum_{k=0}^{C-1}\Delta_{ik}S_k(M)\right)\right]E\left[\left(\sum_{l=0}^{C-1}\Delta_{il}S_l(M)\right)\right] \\
&= \sum_{k=0}^{C-1}\sum_{l=0}^{C-1}\Delta_{ik}\Delta_{jl}E[S_k(M)S_l(M)] - \sum_{k=0}^{C-1}\sum_{l=0}^{C-1}\Delta_{ik}\Delta_{jl}E[S_k(M)]E[S_l(M)] \\
&= \sum_{k=0}^{C-1}\sum_{l=0}^{C-1}\Delta_{ik}\Delta_{jl}(E[S_k(M)S_l(M)] - E[S_k(M)]E[S_l(M)]) \\
&= \sum_{k=0}^{C-1}\sum_{l=0}^{C-1}\Delta_{ik}\Delta_{jl}M\left((M-1)q_{kl}(M) + q_k(M)I_{\{k=l\}} - Mq_k(M)q_l(M)\right). \qquad (3.41)
\end{aligned}
$$

In an identical manner we can also derive the covariance of $W_i'(M)$ and $W_j'(M)$ as

$$
\begin{aligned}
&E[W_i'(M)W_j'(M)] - E[W_i'(M)]E[W_j'(M)] \\
&= \sum_{k=0}^{C-1}\sum_{l=0}^{C-1}\Delta_{ik}\Delta_{jl}M\left((M-1)q'_{kl}(M) + q'_k(M)I_{\{k=l\}} - Mq'_k(M)q'_l(M)\right). \qquad (3.42)
\end{aligned}
$$

Finally, we can simplify these expressions further if we know we are in the set $\mathcal{L}\cap\mathcal{D}$. In this case we can write

$$
\begin{aligned}
&E[W_i(M)W_j(M)] - E[W_i(M)]E[W_j(M)] \\
&= E[(\Delta_{ii}S_i(M) + \delta_i(M - S_i(M)))(\Delta_{jj}S_j(M) + \delta_j(M - S_j(M)))] \\
&\quad - E[\Delta_{ii}S_i(M) + \delta_i(M - S_i(M))]E[\Delta_{jj}S_j(M) + \delta_j(M - S_j(M))] \\
&= E[(\tau_iS_i(M) + M\delta_i)(\tau_jS_j(M) + M\delta_j)] - E[\tau_iS_i(M) + M\delta_i]E[\tau_jS_j(M) + M\delta_j] \\
&= \tau_i\tau_jE[S_i(M)S_j(M)] - \tau_i\tau_jE[S_i(M)]E[S_j(M)] \\
&= \tau_i\tau_jM\left((M-1)q_{ij}(M) + q_i(M)I_{\{i=j\}} - Mq_i(M)q_j(M)\right) \qquad (3.43)
\end{aligned}
$$

and

$$\begin{aligned} & E[W_i'(M)W_j'(M)] - E[W_i'(M)]E[W_j'(M)] \\ &= \tau_i\tau_j M\left((M-1)q_{ij}'(M) + q_i'(M)I_{\{i=j\}} - Mq_i'(M)q_j'(M)\right). \end{aligned} \tag{3.44}$$

Hence, we see that we can also solve for covariances as long as we can find both $q_i(M)$ and $q_{ij}(M)$. By the previous section, at the present this is feasible whenever we are in a constant weight category.

3.5 Examples

In this section we present some applications of the finite memory generalized urn model. The first example is from the field of information theory, an appropriate application since finite urns first appeared in this context. The next major application we consider shows how the functional weight finite urn in its present form can even be used as a reasonable model for a certain class of ant-based routing problem.

3.5.1 Binary Channels with Memory

In [1], Alajaji and Fuja introduced a finite memory contagion channel (FMCC) model for a binary channel with memory. A generalization of this model based on a finite queue, the queue-based channel (QBC), has recently been proposed in [51]. These models have the advantage of being more mathematically tractable compared to standard binary channel models with memory, such as the Gilbert-Elliot channel model

[27], [23] and the Fritchman channel model [25], both of which are based on a hidden Markov structure. The generalized finite urn model considered in this chapter with $C = 2$ colours provides another generalization of the FMCC. In this example we explore this new channel model and its usefulness in terms of deriving statistical and information theoretical quantities of interest, such as bit error rate (BER), correlation between two consecutive noise bits (Cor), block transition probabilities, and capacity.

We label the $C = 2$ colours 0 and 1, where 0 represents no error and 1 represents an error. The error for the n^{th} bit, Z_n, is given by the colour label of the n^{th} draw. We refer to the underlying Markov process with 2^M states, that is $X(n, M)$, as the *state process*. Throughout this example we consider only the class of models for which we have the stationary distribution of the state process in closed form. Recall, this is the class of exchangeable, definite urns or $\mathcal{D}$ in Figure 3.1. We assume that the initial weights, α_0 and α_1 are both zero for simplicity, and that the initial distribution of the state process is the stationary distribution. Thus, $\{Z_n\}_{n=1}^{\infty}$ is a stationary, ergodic, M^{th} order Markov process, which is a key to mathematical tractability.

The channels under consideration are binary additive channels. Specifically, if X_n is the n^{th} input bit to the channel and Z_n is the n^{th} noise bit, the n^{th} output of the channel is $Y_n = X_n \oplus Z_n$, where $\oplus$ denotes addition modulo 2. We assume that X_n and Z_n are independent for all n.

First, we consider linear weight functions represented by the weight matrix

$$\Delta = \begin{bmatrix} \Delta_{00} & \delta_0 \\ \delta_1 & \Delta_{11} \end{bmatrix}.$$

This implies that

$$g_0(x) = M\delta_0 + (\Delta_{00} - \delta_0)s_0(x) = M\delta_0 + \tau_0 s_0(x) \tag{3.45}$$

and

$$g_1(x) = M\delta_1 + (\Delta_{11} - \delta_1)s_1(x) = M\delta_1 + \tau_1 s_1(x). \tag{3.46}$$

We will consider different sets of functions g_0 and g_1 later in this example, but for now let us explore this specific update mechanism. In fact, for the moment assume that $\tau_0 = \tau_1 = \tau$, which says that Δ has constant column sums. In this case the functional weight finite urn is in the set $\mathcal{L} \cap \mathcal{D} \cap \mathcal{C}$ and we have seen that the normalizing constant of the stationary distribution can be computed explicitly. The stationary distribution of the state process is given by

$$\begin{aligned} \pi_{X,x}(M) &= \frac{\prod_{l_0=0}^{s_0(x)-1}(M\delta_0 + l_0\tau)\prod_{l_1=0}^{s_1(x)-1}(M\delta_1 + l_1\tau)}{\prod_{l=0}^{M-1}(M(\delta_0+\delta_1) + l\tau)} \\ &= \frac{\prod_{l_0=0}^{s_0(x)-1}(\sigma + k\delta)\prod_{l_1=0}^{s_1(x)-1}(\rho + j\delta)}{\prod_{l=1}^{M-1}(1 + l\delta)}, \end{aligned}$$

where we have used $\rho = \delta_1/(\delta_0 + \delta_1)$, $\sigma = 1 - \rho$ and $\delta = \tau/(M(\delta_0 + \delta_1))$ for easy comparison with [1]. If $M\delta_0$, $M\delta_1$ and τ are all positive integers, this stationary distribution is identical to the stationary distribution of the finite memory Pólya urn process described in [1], where the urn begins with $M\delta_0$ black balls and $M\delta_1$ red balls, and each time a ball is selected τ additional balls of the selected colour are added to the urn while the balls that were added M draws previously are removed from the urn. Indeed, the probability of selecting a colour 0 ball in state x is

$$\begin{aligned} p_{x,\tilde{x}_{(0)}}(M) &= \frac{M\delta_0 + \tau s_0(x)}{M\delta_0 + \tau s_0(x) + M\delta_1 + \tau s_1(x)} \\ &= \frac{M\delta_0 + \tau s_0(x)}{M(\delta_0 + \delta_1) + M\tau} \\ &= \frac{\delta_0/(\delta_0 + \delta_1) + s_0(x)\tau/(M(\delta_0 + \delta_1))}{1 + M\tau/(M(\delta_0 + \delta_1))} \\ &= \frac{\sigma + s_0(x)\delta}{1 + M\delta}. \end{aligned}$$

Similarly, the probability of selecting a colour 1 ball in state x is

$$p_{x,\tilde{x}_{(1)}}(M) = \frac{\rho + s_1(x)\delta}{1 + M\delta}.$$

These are the same as the transition probabilities for the urn model underlying the FMCC in [1]. Thus, when $\tau_0 = \tau_1 = \tau$ this channel is equivalent to the FMCC, with identical statistical and information theoretical properties.

Allowing $\tau \geq 0$ to be an arbitrary nonnegative real number is a straightforward extension of the FMCC, requiring only a reinterpretation of τ as a weight, rather than as a number of balls, that gets added to the urn. We note that the case $\tau = 0$ corresponds to the memoryless binary symmetric channel (BSC) with crossover probability ρ. The case $\tau < 0$ is also possible. One must constrain τ so that the urn never has a negative weight for either ball colour. The parametrization here in terms of the weight matrix Δ ensures this. However, the case $\tau < 0$ corresponds to an error free bit making an error on the subsequent bit more likely, and vice-versa, so may be of limited interest. In the following we consider several more useful generalizations of the FMCC.

Generalized Finite Memory Contagion Channel

Now consider the more general case of $\tau_0 \neq \tau_1$, with $\tau_0 > 0$ and $\tau_1 > 0$. This is now a 4 parameter model with parameters $\rho = \delta_1/(\delta_0 + \delta_1)$, $\epsilon_0 = \tau_0/(M(\delta_0 + \delta_1))$, $\epsilon_1 = \tau_1/(M(\delta_0+\delta_1))$, and M. Again let $\sigma = 1-\rho$. This model allows differential weighting of the colour 0 balls and the colour 1 balls and so provides for added flexibility in modelling the contagion rate of errors. We will refer to this error process as a generalized finite memory contagion channel (GFMCC). The transition probabilities for the underlying state process are given by

$$p_{x,\tilde{x}_{(0)}}(M) = \frac{\sigma + s_0(x)\epsilon_0}{1 + s_0(x)\epsilon_0 + s_1(x)\epsilon_1}$$

and

$$p_{x,\tilde{x}_{(1)}}(M) = \frac{\rho + s_1(x)\epsilon_1}{1 + s_0(x)\epsilon_0 + s_1(x)\epsilon_1}.$$

The stationary distribution of the state process is given by

$$\pi_{X,x}(M) = G(M)^{-1}(1 + s_0(x)\epsilon_0 + s_1(x)\epsilon_1) \prod_{l_0=0}^{s_0(x)-1} (\sigma + l_0\epsilon_0) \prod_{l_1=0}^{s_1(x)-1} (\rho + l_1\epsilon_1),$$

where

$$\begin{aligned} G(M) &= \sum_x (1 + s_0(x)\epsilon_0 + s_1(x)\epsilon_1) \prod_{l_0=0}^{s_0(x)-1} (\sigma + l_0\epsilon_0) \prod_{l_1=0}^{s_1(x)-1} (\rho + l_1\epsilon_1) \\ &= \sum_{\omega=0}^{M} \binom{M}{\omega} (1 + (M-\omega)\epsilon_0 + \omega\epsilon_1) \prod_{l_0=0}^{M-\omega-1} (\sigma + l_0\epsilon_0) \prod_{l_1=0}^{\omega-1} (\rho + l_1\epsilon_1) \end{aligned}$$

is the normalizing constant. To our knowledge, $G(M)$ does not simplify further in this case. We also suspect that the sequence $\{Z_1, \ldots, Z_M\}$, which is a finite exchangeable

sequence, is not embeddable in an infinite exchangeable sequence unless $\epsilon_0 = \epsilon_1$. As a consequence marginal distributions of $\{Z_1, \ldots, Z_M\}$ should be computed as appropriate sums, which unfortunately do not simplify. We remark, however, that none of this lack of simplification presents any computational burden. In the following we provide finite sum expressions for the bit error rate (BER), the correlation between two consecutive noise bits (Cor), the block transition probabilities and the capacity. We will see that the derived capacity expressions are no more complex than those given for the FMCC in [1] or the QBC in [51].

Define

$$L(\omega, M) = G(M)^{-1}(1 + (M-\omega)\epsilon_0 + \omega\epsilon_1) \prod_{l_0=0}^{M-\omega-1} (\sigma + l_0\epsilon_0) \prod_{l_1=0}^{\omega-1} (\rho + l_1\epsilon_1).$$

Assuming $M \geq 2$, the BER of the GFMCC is given by

$$\text{BER} = \sum_{\omega=1}^{M} \binom{M-1}{\omega-1} L(\omega, M),$$

the probability of two consecutive errors is given by

$$\Pr\{Z_1 = 1, Z_2 = 1\} = \sum_{\omega=2}^{M} \binom{M-2}{\omega-2} L(\omega, M)$$

and the correlation between Z_1 and Z_2 is given by

$$\text{Cor} = \frac{\Pr\{Z_1 = 1, Z_2 = 1\} - (\text{BER})^2}{\text{BER}(1 - \text{BER})}.$$

If $M = 1$ then

$$\text{BER} = \Pr\{Z_1 = 1\} = \pi_{X,1}(1) = \frac{(1+\epsilon_1)\rho}{(1+\epsilon_0)\sigma + (1+\epsilon_1)\rho} = \frac{(1+\epsilon_1)\rho}{1 + \epsilon_0\sigma + \epsilon_1\rho},$$

$$\begin{aligned}\Pr\{Z_1 = 1, Z_2 = 1\} &= \pi_{X,1}(1)p_{1,1}(1) \\ &= \left(\frac{(1+\epsilon_1)\rho}{1+\epsilon_0\sigma+\epsilon_1\rho}\right)\left(\frac{\rho+\epsilon_1}{1+\epsilon_1}\right) \\ &= \frac{\rho(\rho+\epsilon_1)}{1+\epsilon_0\sigma+\epsilon_1\rho}\end{aligned}$$

and

$$\text{Cor} = \frac{p_{1,1}(1) - \pi_{X,1}(1)}{1-\pi_{X,1}(1)} = \frac{\rho+\epsilon_1}{1+\epsilon_1} - \frac{\rho(1-\rho)}{(1+\epsilon_0)\sigma}.$$

We now consider the block transition probabilities for this channel. Let $Z_1^n = (Z_1, \ldots, Z_n)$ denote a sequence of n bits from the error process. Let X_1^n denote a sequence of n bits input into the channel and let Y_1^n denote the n bits at the output of the channel corresponding to the bits in X_1^n. For a given input sequence $x_1^n = (x_1, \ldots, x_n) \in \{0,1\}^n$ and output sequence $y_1^n = (y_1, \ldots, y_n) \in \{0,1\}^n$, the block transition probability is

$$\Pr\{Y_1^n = y_1^n | X_1^n = x_1^n\} = \Pr\{Z_1^n = z_1^n\}$$

where $z_1^n = x_1^n \oplus y_1^n$ and $\oplus$ denotes summation modulo 2. That is, the channel is *symmetric* in the sense defined in [1]. If $n > M$ then for $M < i \leq n$ let $d_i = \sum_{j=i-M}^{i-1} z_j$ be the number of 1's in the M-bit block immediately preceding z_i. If $n \leq M$, let $s_1(z_1^n)$ denote the number of 1's in z_1^n. The block transition probabilities are given by the following:

- For blocklength $n \leq M$

$$\Pr\{Z_1^n = z_1^n\} = \sum_{\omega=s_1(z_1^n)}^{s_1(z_1^n)+M-n} \binom{M-n}{\omega - s_1(z_1^n)} L(\omega, M),$$

- For blocklength $n > M$

$$\Pr\{Z_1^n = z_1^n\} = L(d_{M+1}, M) \prod_{i=M+1}^{n} p(d_i, M)^{z_i} (1 - p(d_i, M))^{1-z_i},$$

where

$$p(\omega, M) = \frac{\rho + \omega\epsilon_1}{1 + (M-\omega)\epsilon_0 + \omega\epsilon_1}$$

is the probability of an error, or equivalently the probability of drawing colour type 1, when the state has ω ones.

The capacity of a channel is a fundamental information theoretic quantity introduced by Shannon [42]. It represents the maximum rate at which information can be transmitted over the channel and still be received at the receiver with arbitrarily small probability of error by using channel coding. To achieve capacity one must typically use codes with infinite blocklength, which are impractical, but the channel capacity is nonetheless a key theoretical quantity which serves as a benchmark for constructing good practical codes and for comparing channel models.

As in [1], all the binary channel models with memory considered in this example have additive, stationary, ergodic noise and are symmetric. Furthermore, they are information stable [47], and so we can calculate the capacity as

$$C(M) = 1 - \mathcal{H}^{(M)}(Z),$$

where $\mathcal{H}^{(M)}(Z)$ is the entropy rate of the M^{th} order Markov noise process Z, and is defined as

$$\mathcal{H}^{(M)}(Z) = \lim_{n\to\infty} \frac{1}{n} H(Z_1, \ldots, Z_n),$$

with $H(Z_1, \ldots, Z_n)$ is the joint entropy of $(Z_1, \ldots, Z_n)$. Moreover, for a stationary process such as Z we can replace $\frac{1}{n}H(Z_1, \ldots, Z_n)$ in the entropy rate above with the conditional entropy $H(Z_n|Z_{n-1}, \ldots, Z_1)$ and, since the noise process Z is a time homogeneous M^{th} order Markov process we may write $H(Z_n|Z_{n-1}, \ldots, Z_1) = H(Z_{M+1}|Z_M, \ldots, Z_1)$. Note that definitions of joint entropy and conditional entropy can be found in any textbook on information theory, for example [14]. The computation of the capacity here is the same as in [1] and [51], so the reader is referred to these papers for further details.

We have

$$\begin{aligned}
\mathcal{H}^{(M)}(Z) &= \lim_{n\to\infty} \frac{1}{n} H(Z_1, \ldots, Z_n) \\
&= H(Z_{M+1}|Z_M, \ldots, Z_1) \\
&= H(X(2, M)|X(1, M)) \\
&= -\sum_{x,\tilde{x}} \pi_{X,x}(M) p_{x,\tilde{x}}(M) \log_2 p_{x,\tilde{x}}(M) \\
&= \sum_{\omega=0}^{M} \binom{M}{\omega} L(\omega, M) h_b(p(\omega, M)),
\end{aligned}$$

where $h_b(p) = -p\log_2 p - (1-p)\log_2(1-p)$ is the binary entropy function. Thus, the capacity of the GFMCC is

$$C(M) = 1 - \sum_{\omega=0}^{M} \binom{M}{\omega} L(\omega, M) h_b(p(\omega, M)).$$

An important observation to make here is that the capacity expression above is of the same form as that given in [1] for the finite memory contagion channel. It differs only in $L(\omega, M)$ and $p(\omega, M)$. Indeed, the capacity for all the channels considered in this example have this form. In fact, the expressions for BER, $\Pr\{Z_1 = 1, Z_2 = 1\}$, Cor, and the block transition probabilities for all the channels in this example are the same as the expressions above for the GFMCC provided appropriate expressions for $L(\omega, M)$ and $p(\omega, M)$ are substituted. Thus, for subsequent channel descriptions we will only provide the corresponding expressions for $L(\omega, M)$ and $p(\omega, M)$.

Power Law Contagion Channel

We have seen that the probability of an error when the state has ω ones in the case when $\tau_0 = \tau_1 = \tau$ is a linearly increasing function of ω. One way to interpret the generalization $\tau_0 \neq \tau_1$ is to consider how it generalizes $p(\omega, M)$. When $\tau_0 \neq \tau_1$, $p(\omega, M)$ is a nonlinear function of ω because of the non-constant denominator. In fact, if $\tau_1 < \tau_0$ the rate of increase is superlinear and the contagion accelerates as the state gets more ones. If $\tau_1 > \tau_0$ the rate of increase is sublinear so the contagion decelerates as the state gets more ones. We may model superlinear or sublinear contagion more directly without changing the total weight of any state by defining appropriate weight functions $g_0(x)$ and $g_1(x)$.

Let

$$g_1(x) = p_{min} + \frac{p_{max} - p_{min}}{M^r} s_1(x)^r \tag{3.47}$$

and

$$g_0(x) = 1 - g_1(x), \tag{3.48}$$

where $0 < p_{min} \leq p_{max} < 1$ and $r > 0$. With $\alpha_0 = \alpha_1 = 0$, the weight of every state is constant and equal to 1. When the state has ω ones the next draw is a one with probability

$$p(\omega, M) = p_{min} + \frac{p_{max} - p_{min}}{M^r} \omega^r.$$

When $p_{min} = p_{max}$ the channel simplifies to a BSC with crossover probability p_{min}. We assume henceforth that $p_{min} < p_{max}$. When $r = 1$, $p(\omega, M)$ increases linearly from p_{min} to p_{max} and so corresponds to the FMCC. When $r > 1$ the rate of increase is superlinear and when $0 < r < 1$ the rate of increase is sublinear. This is a 4 parameter functional weight finite urn channel model, with parameters p_{min}, $p_\Delta = p_{max} - p_{min}$, r and M. We shall refer to this model as the power law contagion channel (PLCC). Note that because of the different weight functions this channel is not equivalent to the GFMCC. It is true however, that both the GFMCC and the PLCC each have the FMCC as a special case.

This time the stationary distribution of state x is given by

$$\pi_{X,x}(M) = G(M)^{-1} \prod_{l_0=0}^{s_0(x)-1} \left(1 - p_{min} - \frac{p_\Delta}{M^r}(M - l_0)^r\right) \prod_{l_1=0}^{s_1(x)-1} \left(p_{min} + \frac{p_\Delta}{M^r} {l_1}^r\right),$$

where

$$G(M) = \sum_{\omega=0}^{M} \binom{M}{\omega} \prod_{l_0=0}^{M-\omega-1} \left(1 - p_{min} - \frac{p_\Delta}{M^r}(M - l_0)^r\right) \prod_{l_1=0}^{\omega-1} \left(p_{min} + \frac{p_\Delta}{M^r} {l_1}^r\right).$$

Finally, we have

$$L(\omega, M) = G(M)^{-1} \prod_{l_0=0}^{M-\omega-1} \left(1 - p_{min} - \frac{p_\Delta}{M^r}(M - l_0)^r\right) \prod_{l_1=0}^{\omega-1} \left(p_{min} + \frac{p_\Delta}{M^r} {l_1}^r\right).$$

Self Regulating Gilbert-Elliot Channel

We now consider a generalized finite urn model, once again with $C = 2$, which makes draws according to two binary symmetric channels (BSCs), alternating between them. We shall refer to this model as a self regulating Gilbert-Elliot channel (SRGEC). For a given memory order M define a threshold parameter $T \in \{1, \ldots, M\}$. Also define p_G and p_B, with $p_G < p_B$, to be the crossover probabilities for the two BSCs. When the state x is such that $s_1(x) < T$ then we draw a 0 with probability $1 - p_G$ and a 1 with probability p_G. That is, we draw according to the *good* BSC. When the state x is such that $s_1(x) \geq T$ then we draw a 0 with probability $1 - p_B$ and a 1 with probability p_B. That is, we draw according to the *bad* BSC. This is the functional weight finite urn model with 2 colours and weight functions given by

$$g_0(x) = \begin{cases} 1 - p_G & \text{if } s_0(x) > M - T \\ 1 - p_B & \text{if } s_0(x) \leq M - T \end{cases} \tag{3.49}$$

and

$$g_1(x) = \begin{cases} p_G & \text{if } s_1(x) < T \\ p_B & \text{if } s_1(x) \geq T \end{cases}. \tag{3.50}$$

These update functions are clearly of a different form than any of those considered thus far. Indeed, these expressions do not even reduce to a linear form, as is characteristic of the FMCC. It is evident that they correspond to yet another unique binary channel arising from our functional weight finite urn. In fact, we see that there are an infinite number of different channels that we may propose due to the general nature of our update functions.

Notice that the total weight of any state x is 1. Moreover, the probability of an error when there are ω ones is

$$p(\omega, M) = \begin{cases} p_G & \text{if } \omega < T \\ p_B & \text{if } \omega \geq T \end{cases}$$

and the stationary distribution of the state process is

$$\begin{aligned} \pi_{X,x}(M) &= \begin{cases} G(M)^{-1} p_G^{s_1(x)} (1-p_G)^{T-s_1(x)-1} (1-p_B)^{M-T+1} & \text{if } s_1(x) < T \\ G(M)^{-1} p_G^{T} p_B^{s_1(x)-T} (1-p_B)^{M-s_1(x)} & \text{if } s_1(x) \geq T \end{cases} \\ &= \begin{cases} G(M)^{-1} \left(\frac{1-p_B}{1-p_G}\right)^{M-T+1} p_G^{s_1(x)} (1-p_G)^{M-s_1(x)} & \text{if } s_1(x) < T \\ G(M)^{-1} \left(\frac{p_G}{p_B}\right)^{T} p_B^{s_1(x)} (1-p_B)^{M-s_1(x)} & \text{if } s_1(x) \geq T \end{cases} \end{aligned}$$

where the normalizing constant $G(M)$ is given by

$$\begin{aligned} & G(M) \\ &= \left(\frac{1-p_B}{1-p_G}\right)^{M-T+1} \sum_{\omega=0}^{T-1} \binom{M}{\omega} p_G^{\omega} (1-p_G)^{M-\omega} + \left(\frac{p_G}{p_B}\right)^{T} \sum_{\omega=T}^{M} \binom{M}{\omega} p_B^{\omega} (1-p_B)^{M-\omega} \\ &= \left(\frac{1-p_B}{1-p_G}\right)^{M-T+1} \Pr\{B(M, p_G) < T\} + \left(\frac{p_G}{p_B}\right)^{T} \Pr\{B(M, p_B) \geq T\}, \end{aligned}$$

and $B(M,p)$ denotes a Binomial(M,p) random variable.

The quantity $L(\omega, M)$ can be expressed as

$$L(\omega,M) = \begin{cases} G(M)^{-1}\left(\frac{1-p_B}{1-p_G}\right)^{M-T+1} p_G^{\omega}(1-p_G)^{M-\omega} & \text{if } \omega < T \\ G(M)^{-1}\left(\frac{p_G}{p_B}\right)^{T} p_B^{\omega}(1-p_B)^{\omega} & \text{if } \omega \geq T \end{cases}.$$

The simplest example of the SRGEC is when $M = T = 1$. For every bit in error the subsequent noise bit is drawn according to the bad BSC while for every error free bit the subsequent noise bit is drawn according to the good BSC. Thus, the burstiness of the channel noise is determined by the parameters p_G and p_B of the BSCs themselves. The noise process is simply a two state Markov chain with transition matrix

$$P = \begin{bmatrix} 1-p_G & p_G \\ 1-p_B & p_B \end{bmatrix}$$

and corresponds to a Fritchman$(1,1)$ channel. In another particular case, when $M = T = 2$, a switch from the good BSC to the bad BSC occurs only when two consecutive errors occur in the good BSC while a switch from the bad BSC to the good BSC occurs as soon as an error free bit is drawn in the bad BSC.

We have considered several extensions to the FMCC based on the functional weight finite urn. We have seen that the BER, Cor, block transition probabilities and capacity of these channels admit closed form, easily computable expressions. Our investigation into these channel models is preliminary, as this is not the main topic of this work. For a more in depth treatment of the kind of analysis and application that we believe can be applied to our channel models see the recent work on the QBC of Zhong, et al. [51], [52], [54] and the Ph.D. thesis of Zhong [53].

3.5.2 Static Ant Routing

In this example we use a functional weight finite urn to model an ant routing algorithm. Our view is focused at a particular node of the routing network. We will assume that the colours of the urn correspond to possible routes out of this local node. We take the probability of choosing route i, $0 \leq i \leq C-1$, at any time according to the distribution given by the current urn composition.

At this stage, the only feedback we have to work with is the history of selections made at the local node. That is, in any real implementation we expect that feedback at a node reflects the experience on the entire route, not just the decision at the first hop, but this ideal information is not available in the current model. Still we can set our urn parameters to mimic the type of behaviour we want to study. We know that as the local node sends more and more traffic on the same route, the potential exists to saturate that route with detrimental consequences. It is better that the algorithm detect this situation and throttle back the traffic before overwhelming a single route. In fact, a simple ant routing algorithm may instruct each node to behave in this way in order to achieve reasonable load balancing without the complexity of inter-node message passing.

Based on our discussion so far, a linear form urn is inappropriate. We want each functional component of the weight to initially increase with increasing decisions in favour of a given route, and then later taper off to perform the throttling we have mentioned. A linear g_i cannot accomplish this satisfactorily. Still, insisting that each g_i is a function of only the number of slots of colour i in a given state is not too much of a restriction. Remaining in this definite case allows us to use the stationary distribution form result. From this we can calculate means and covariances as basic

measures of how the system is performing.

Suppose now we outline a small and simple specific instance of the type of urn we have described. Take $C = 2$ and $M = 5$. Further, imagine

$$g_i(x) = a_i\sqrt{s_i(x)} \tag{3.51}$$

for each i, where we have $x = (x_0, x_1, x_2, x_3, x_4)$ and $s_i(x) = \sum_{k=0}^{4} I_{\{x_k=i\}}$ so that $s_i(x)$ counts the number of slots of colour i in state x as usual. Note that the parameter a_i allows us to differentiate the routes. The square root function is a simple, concave function which is easily parameterizable. Since the feedback here is deterministic, we must gauge our preference of routes, perhaps due to some physical feature that does not vary, ahead of time. Without loss of generality, we will assume that route 0 is preferred to route 1 by insisting that $a_0 \geq a_1$ so that the weight given to route 0 is always at least as much as that given to route 1 for the same number of slots occupied.

Now, the weight associated with route i when the urn is in state x is given by

$$w_i(x) = \alpha_i + a_i\sqrt{s_i(x)}, \tag{3.52}$$

and the total weight is $w(x) = w_0(x)+w_1(x)$. With these definitions we can determine all transition probabilities for this process.

In order to analyze this urn we first use Lemma 3.1 to construct

$\pi_{S,s}(5)$

$$
\begin{aligned}
&= G(5)^{-1}\binom{M}{s_0, s_1}\left(\sum_{k=0}^{2-1}(\alpha_k + a_k\sqrt{s_k})\right)\left(\prod_{l_0=0}^{s_0-1}\left(\alpha_0 + a_0\sqrt{l_0}\right)\right)\left(\prod_{l_1=0}^{s_1-1}\left(\alpha_1 + a_1\sqrt{l_1}\right)\right) \\
&= G(5)^{-1}\binom{M}{s_0}\left(\sum_{k=0}^{1}(\alpha_k + a_k\sqrt{s_k})\right)\left(\prod_{l_0=0}^{s_0-1}\left(\alpha_0 + a_0\sqrt{l_0}\right)\right)\left(\prod_{l_1=0}^{5-s_0-1}\left(\alpha_1 + a_1\sqrt{l_1}\right)\right)
\end{aligned}
$$

where

$$
\begin{aligned}
&G(5) \\
&= \sum_s \binom{5}{s_0, s_1}\left(\sum_{k=0}^{2-1}(\alpha_k + a_k\sqrt{s_k})\right)\left(\prod_{l_0=0}^{s_0-1}\left(\alpha_0 + a_0\sqrt{l_0}\right)\right)\left(\prod_{l_1=0}^{s_1-1}\left(\alpha_1 + a_1\sqrt{l_1}\right)\right) \\
&= \sum_{s_0=0}^{5}\binom{5}{s_0}\left(\sum_{k=0}^{1}(\alpha_k + a_k\sqrt{s_k})\right)\left(\prod_{l_0=0}^{s_0-1}\left(\alpha_0 + a_0\sqrt{l_0}\right)\right)\left(\prod_{l_1=0}^{5-s_0-1}\left(\alpha_1 + a_1\sqrt{l_1}\right)\right).
\end{aligned}
$$

Once we have found $\pi_{S,s}(5)$ we can calculate things like the probability of selecting route i, previously understood to be $p_i(5) = q_i(5)$. This is simply

$$p_i(5) = q_i(5) = E[W_i(5)/W(5)] = \sum_s \frac{w_i(s)}{w(s)}\pi_{S,s}(5).$$

We want to see that the probability of drawing colour i increases if we adjust the parameters so that route i is favoured. We must ensure however, that any increase in $p_i(5)$ is not accompanied by instability. To this end we also want a measure of the variability of $p_i(5)$. We see that $p_i(5)$ is actually the expected value of the random variable $W_i(5)/W(5)$. We simply calculate the variance of this random variable as

$$
\begin{aligned}
&E[W_i(5)/W(5)W_i(5)/W(5)] - E[W_i(5)/W(5)]E[W_i(5)/W(5)] \\
&= \sum_s \frac{w_i(s)}{w(s)}\frac{w_i(s)}{w(s)}\pi_{S,s}(5) - \left(\sum_s \frac{w_i(s)}{w(s)}\pi_{S,s}(5)\right)\left(\sum_s \frac{w_i(s)}{w(s)}\pi_{S,s}(5)\right).
\end{aligned}
$$

More than likely, the goal may actually be to study the relationship between a combination of α_0, α_1, a_0 and a_1 and some resulting measure like p_i above. Moreover, a_0 and a_1 parametrize g_0 and g_1 respectively, but we may be interested in studying the effect of various functional weight components in designing an ant algorithm.

As a concrete illustration of this technique we isolate a single parameter and study its effect on $p_i(5)$. Suppose $\alpha_0 = \alpha_1 = 1$ and $a_1 = 3$, with the latter value selected so that even in the worst case we have a $\frac{1}{2+3\sqrt{5}}$ chance of choosing the preferred route, which is route 0 here. Now consider a range of values of a_0. Figure 3.2 plots $p_0(5)$ versus a_0 for $3 \leq a_0 \leq 12$ in steps of 0.5. Note that since there are only two colours in this example, and our measure is a probability, our plot actually gives a complete picture of $p_1(5)$ indirectly as well.

Observe immediately that when $a_0 = a_1 = 3$ the probability of drawing either colour is equal, as it must be by symmetry. Also, notice that the curve appears to remain away from the line $p_0(5) = 1$. This too is expected, since the base weight ensures that there is always a chance a currently unpreferred colour type is drawn. Since we are not adjusting the base weights to reflect the fact that the total weight of each state is growing as a_0 is increased, eventually $p_0(5)$ would become arbitrarily close to 1, but in this small range, our simple approach is sufficient. The main thing we are looking for in fact, pertains more to the shape and rate of increase of the curve. In a more complete ant algorithm design analysis we would perhaps compare this curve's characteristics to other potential weight functions.

Next, in Figure 3.3 we consider the stability of our primary measure $p_0(5)$.

Overall, it seems that the variability actually decreases as we increase the parameter reflecting the preference of route 0. This is an ideal situation. This initial analysis

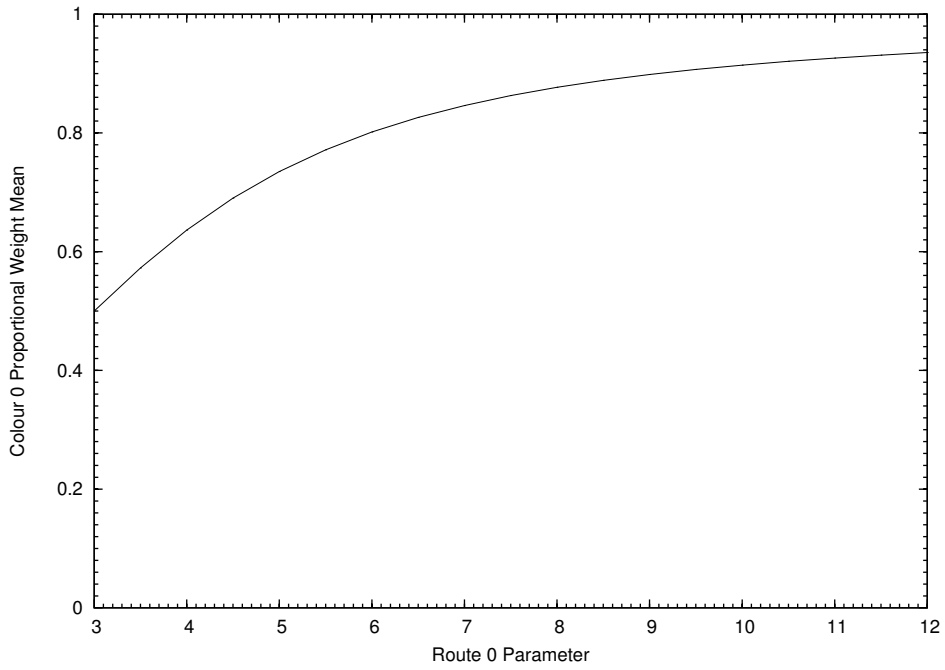

Figure 3.2: Probability of drawing colour 0 versus parameter a_0.

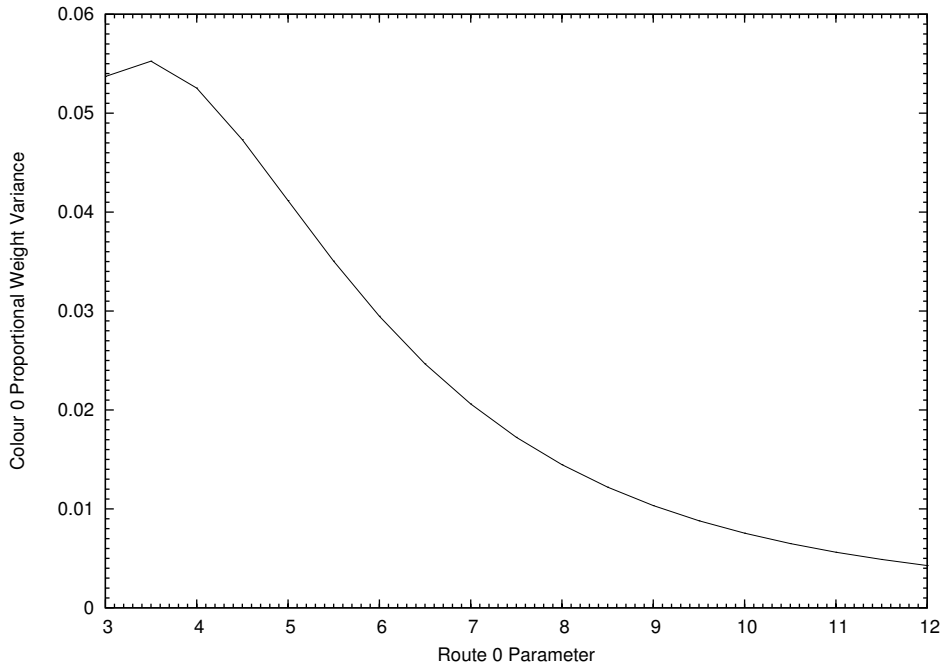

Figure 3.3: Variance of the probability of drawing colour 0 versus parameter a_0.

shows us here that it is safe to increase a_0, so that the algorithm can more easily determine the best route, without worrying that much undesirable instability is introduced to the system.

3.5.3 Other Applications

Urns have found many applications as probabilistic models of random phenomena. With the generalizations above we have presented a finite urn that is essentially on par with more traditional urns in terms of flexibility. The remaining difference, finiteness itself, at the very least adds a new twist to previous applications. In some cases, the finiteness element is a crucial advantage. For example, recall the reason we use a finite urn as a basis for modelling ant algorithms. We want a way in which to model evaporation of pheromone. With an infinite urn the effect of updates becomes diminished over time, but this is not true for ant algorithms. Other applications potentially have this requirement as well.

Consider what we call a user preference model. In this type of system a user is confronted with C different servers, each with individual advantages and drawbacks. We use the term server loosely here. Service may be purchasing a specific object for example. Before making a decision about which service to undergo the user has access to a database of statistics describing the decisions of previous users. It is assumed that this information influences the decision of the current user. In turn, after receiving service, certain aspects of the current decision are logged in the database for future users. So that information in the database does not get stale, information in the database is regularly purged as well. We note here that a given decision has deterministic feedback. In the context of this example, the functional weight finite urn

model is not general enough to allow the possibility of users creating their own reviews for the database. This would require modelling more random feedback, something we consider later. Still there are many non trivial settings in which the existing model can apply - basically any time the process of providing the feedback is automated. Perhaps we can think of a scenario like the basic working of an on-line search engine. Addressable objects may be displayed to a user in a ranked order in which the more users viewing an object increases its ranking. This type of self reinforcement can be adequately modelled.

Another class of applications to consider are multitype branching processes. In the literature, urn analysis often hinges on viewing the urn as a multitype branching process [2], [3], [34]. Drawing a colour type from the urn and returning balls of various colour types corresponds to leaves of a multitype branching process splitting and generating new leaves of various types. The growth of the urn or tree, often referred to as the population, can then be monitored at each time step. Discrete time usually counts one generation to the next. Indeed, urns were originally introduced as a way of studying the behaviour of a population over time. For infinite urns where balls are never removed, or equivalently branching processes where the tree never loses leaves, the population grows without bound. It is not hard to imagine situations in which this is not realistic. Certain applications might be better modelled with a population size that fluctuates within a given range. Staying with the same basic interpretation of the relationship between urns and branching processes, we can contemplate the effect of substituting a finite urn. Corresponding to a finite urn with memory M we can associate a multitype branching process that evolves as before except that members of the population only live for M generations. This

modification opens up several modelling opportunities.

3.6 Discussion

In this chapter we have described a flexible stochastic model, one that we have termed a functional weight finite urn. The model originates out of the literature on Pólya urns, and specifically, follows the relatively young branch of finite urns. We view the evolution of our urn differently, however. Although we are still interested in the composition of the urn in terms of the weight of each colour in the urn, we focus primarily on the succession of draws that lead to the evolution. With this information we allow general functional updates to the urn based on the past history of draws.

At this point we have described the model in detail and investigated some of its properties. We have shown the form of the stationary distribution in certain cases, for example. Moreover, for a fixed memory size M, we have derived expressions for the first and second moments of our stationary urn process. Because we consider the steady state of the urn, we have already begun to contemplate the behaviour of our urn as time goes to infinity, a common topic in urn literature. In the next chapter we will address another important type of limiting behaviour. We will consider the moments as $M \to \infty$.

Chapter 4

Limiting Behaviour

In Chapter 3 we have described a finite urn. Only the last fixed number of draws affect the composition of these types of urns. In our notation this fixed memory size is M. It is reasonable to wonder what occurs as $M \to \infty$. Note that although M can be interpreted in terms of time, we can also think of it as confining the state space of our process. Hence, $M \to \infty$ is different from the time limiting behaviour of traditional urns. In our context, the latter type of limit is $n \to \infty$. So that no confusion arises we will call the limit involving M the spatial limit and the limit involving n the time limit. We will explore these various limits here, for urns in the linear exchangeable set.

In our defined functional weight finite urn process, with proper scaling we can consider either the spatial or the time limit, and in either order. In the first case, we let $M \to \infty$, then scale by n and take $n \to \infty$. That is, recalling that

$$W(n, M) = (W_0(n, M), W_1(n, M), \ldots, W_{C-1}(n, M))$$

is the vector-valued process of weights of each colour, the first order we consider is

$$\lim_{n\to\infty} \frac{1}{\sqrt{n}} \left(\lim_{M\to\infty} \left(W(n,M) - E[W(n,M)) \right) \right) = \lim_{n\to\infty} \frac{W(n,\infty) - E[W(n,\infty)]}{\sqrt{n}}.$$

But we see that letting $M \to \infty$ first, we effectively create an infinite urn, represented here by $W(n,\infty)$, so this procedure is actually equivalent to investigating the time limiting behaviour of a classic scaled urn. This limit has been studied by several authors in various settings. Janson[34], [35], in particular, pulls together a good summary of these infinite urn results and establishes many new results. In [34], the author assumes a very general ball addition matrix, and it is shown that there are several possibilities for the limiting covariance matrix, depending on the eigenvalues of the ball addition matrix Δ. To our knowledge the order of limits where we first let $n \to \infty$ and then later contemplate $M \to \infty$ has not been previously considered. In a sense we are in an ideal position to consider this second order. We have already thoroughly investigated the stationary distribution for fixed M. With the proper scaling we might be able to look at the sequence of stationary distributions, indexed by the parameter M, in the limit. Specifically, this second ordering we consider, which scales by M in place of n in the outer limit, is

$$\lim_{M\to\infty} \frac{1}{\sqrt{M}} \left(\lim_{n\to\infty} \left(W(n,M) - E[W(n,M)] \right) \right) = \lim_{M\to\infty} \frac{W(\infty,M) - E[W(\infty,M)]}{\sqrt{M}},$$

where this time we take $W(\infty, M)$ to represent the stationary process for fixed M that we considered in Chapter 3. It is reasonable, without probing further, to conjecture that the limiting distribution in this second case is the same as for the first type of ordering. That is, it is tempting to assume that

$$\lim_{M\to\infty} \frac{W(\infty,M) - E[W(\infty,M)]}{\sqrt{M}} = \lim_{n\to\infty} \frac{W(n,\infty) - E[W(n,\infty)]}{\sqrt{n}}.$$

If this were true, the limiting behaviour arising from the second ordering would be the same as that characterized in [34]. In particular, for a sequence of finite urns parameterized by a weight matrix satisfying the right conditions, we would expect the limiting distribution of the weights of each colour to be multivariate normal with a known mean vector and covariance matrix. We will see in this chapter, however, that this is not the case.

In Janson's work, the ball addition matrix is classified as either *reducible* or *irreducible*. We also adhere to these terms in order to distinguish various weight matrices. In the following we give a precise definition of this characterization.

For a non-negative ball addition matrix Δ, where we assume each row of Δ has at least one positive entry, let $\Delta_{RS,i} = \sum_{j=0}^{C-1} \Delta_{ij}$ be the sum of the i^{th} row of Δ, and define the matrix $P(\Delta)$ to be Δ with the rows normalized to sum to 1. That is, $P_{ij}(\Delta) = \Delta_{ij}/\Delta_{RS,i}$. The matrix Δ is said to be *irreducible* (*reducible*) if the matrix $P(\Delta)$ is the transition matrix of an irreducible (reducible) Markov chain.

In this work we focus mainly on the irreducible case. The reducible class, at least with regard to infinite urns, is shown to have a variety of limiting behaviours depending on the exact form of the weight matrix [35]. The analogous reducible class of finite urns may similarly divide into categories that each require special treatment.

There is one reducible case however, in which we can already say something about the limiting distribution of a scaled finite urn. This class is characterized by a constant diagonal weight matrix, say $\Delta = \tau I$. In this category, the stationary probability of state s in a finite urn with memory M is, by the results of Chapter 3,

$$\pi_{S,s}(M) = \binom{M}{s_0, \ldots, s_{C-1}} \frac{\prod_{i=0}^{C-1} \left(\prod_{l_i=0}^{s_i-1} (\alpha_i + \tau l_i) \right)}{\prod_{l=0}^{M-1} \left(\sum_{k=0}^{C-1} \alpha_k + \tau l \right)}.$$

But this expression is identical to the probability of drawing s_0 colour 0 balls, s_1 colour 1 balls $, \ldots, s_{C-1}$ colour $C-1$ balls, is some order, in a total of M draws in an infinite urn. Thus, the sequence of stationary distributions, indexed by the memory parameter M, corresponding to a finite urn, is mathematically the same as the sequence of distributions of the urn composition at time M in an infinite urn. But the latter has been studied by many authors. In particular, it is well known that the proportions of each colour type, that is $(S_0, \ldots, S_{C-1})/M$ converges to the Dirichlet distribution with parameters $\alpha_0/\left(\sum_{k=0}^{C-1}\alpha_k\right), \alpha_1/\left(\sum_{k=0}^{C-1}\alpha_k\right), \ldots, \alpha_{C-1}/\left(\sum_{k=0}^{C-1}\alpha_k\right)$, [21]. Beyond this observation however, further investigation of reducible cases in the context of finite urns is left for future work.

In the irreducible case studied in [34] Janson points out that the class of infinite urns subdivides into 3 distinct sets with regard to limiting behaviour. The 3 cases are identified by relationships between the eigenvalues of the weight matrix. The largest, real eigenvalue plays a critical role. Specifically, the magnitude of the second largest eigenvalue is compared to the size of one half the magnitude of the largest eigenvalue and an appropriate scaling factor is applied to acheive a particular limiting result. If the second largest magnitude eigenvalue is less than one half the magnitude of the largest eigenvalue for instance, the centered weight is scaled by $\sqrt{n}$. For the other cases, the exact scaling factors are slightly more complicated, as they depend on additional parameters of the spectral decomposition.

For now, an empirical counter example shows that the scaled limiting distribution of a sequence of our stationary urns as $M \to \infty$ is not the same as the limiting distribution of an analogous infinite urn. That is, imagine a comparison between the distribution $(W_i(M) - E[W_i(M)])/\sqrt{M}$ for large M and the time limiting centred and

scaled distribution of a traditional infinite urn, which we know from the literature is normal. Specifically, consider the parameters $C = 2$, $\alpha = (1, 1)$ and

$$\Delta = \begin{bmatrix} 7 & 1 \\ 5 & 3 \end{bmatrix}.$$

On one hand, we can calculate the distribution of $(W_i(M) - E[W_i(M)])/\sqrt{M}$ exactly for any finite memory using $\pi_X(M)$. On the other hand, suppose we let T_i represent the limiting centred and scaled weight of colour i from traditional infinite urn theory. From [34] we know that in this case we have $(T_0, T_1) \sim Normal(0, \Sigma_T)$, where

$$\Sigma_T = \begin{bmatrix} 18 & 6 \\ 6 & 2 \end{bmatrix}.$$

But consider the plot of the cumulative distribution functions of both $(W_0(10000) - E[W_0(10000)])/\sqrt{10000}$ and T_0 in Figure 4.1.

This plot strongly suggests that these distributions are not the same. The actual variance, as calculated using the stationary distribution with $M = 10000$, is clearly not well approximated by the variance of T_0. Moreover, by repeating this empirical test with different weight matrices it can be shown that the magnitudes of the 2 compared variances are not even always found to be in the same order. Because of this discrepancy, there is an interest in investigating the scaled limiting behaviour of a sequence of stationary finite urns with increasing memory.

For the remainder of this chapter, we consider an irreducible non-negative weight matrix, Δ. We revisit our calculations of values like $E[S_i(M)]$ and $(E[S_i(M)S_j(M)] - E[S_i(M)]E[S_j(M)])$ with the hope of finding meaningful limits of these quantities as

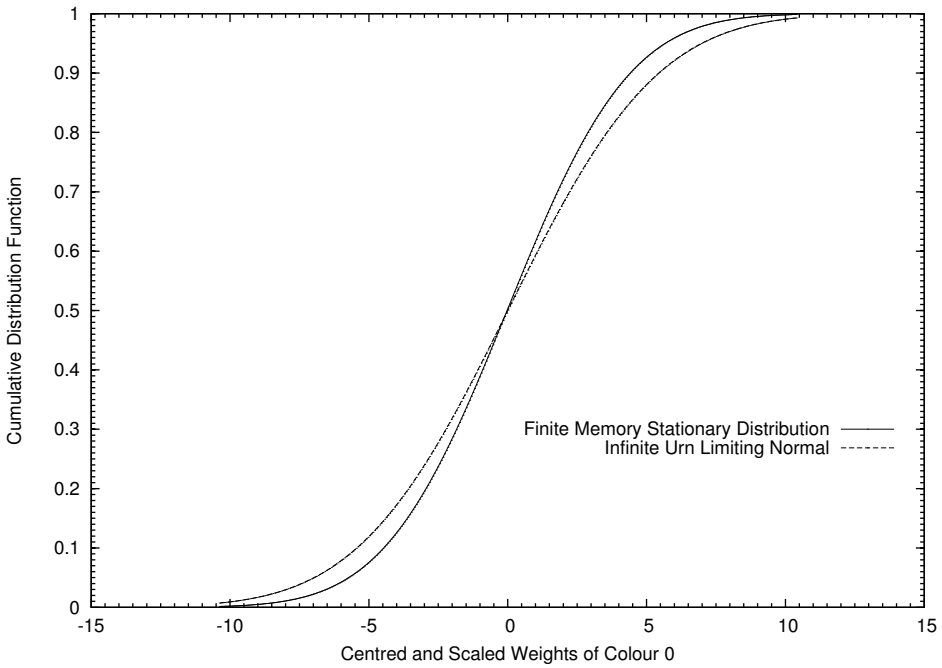

Figure 4.1: Comparison of the Cumulative Distribution Function of the weight of colour 0, as calculated from the stationary distribution with $M = 10000$, and the limiting normal distribution, T_0, of a sequence of infinite urns.

$M \to \infty$. Before we set directly upon this task however, in the next section we first explore a vital relationship between unprimed and primed limiting distributions for our urn.

4.1 Equivalence of Limiting Stationary Systems

Throughout the document up until this point we have carefully defined two separate systems, which we have called an unprimed urn and a primed urn. The key difference between the two is in the presence of a null ball in the latter. We stressed until now that involving two separate systems is for technical reasons, and we have observed that derived expressions often include both unprimed and primed quantities. Of course our main interest remains the unprimed values. In order to solve for these values we need a way to relate primed quantities with their unprimed counterparts. In this section we provide theorems that provide this missing link.

In Section 3.2 we discussed the stationary distributions $\pi_X(M)$ and $\pi_{X'}(M)$ for unprimed and primed urns respectively with fixed M. Here we return to these distributions and consider their relationship to one another for large M. First, using Cov to denote covariance, we define

$$\sigma_{ij}(M) = \frac{\mathrm{Cov}(S_i, S_j)}{M} \quad \text{and} \quad \sigma'_{ij}(M) = \frac{\mathrm{Cov}(S'_i, S'_j)}{M}.$$

We also henceforth assume that $\sigma'_{ij}(M)$ is bounded for all M and in the limit as $M \to \infty$, for all i and j.

Theorem 4.1 *Provided* $\lim_{M\to\infty}\sigma'_{ij}(M)$ *is bounded for all* $0 \le i,j \le C-1$, *in the limit as* $M \to \infty$, *the* $\mathcal{L}^1$ *distance between* $\pi_X(M)$ *and* $\pi_{X'}(M)$ *converges to 0 as* $M \to \infty$. *That is,*

$$\lim_{M\to\infty}\sum_x |\pi_{X,x}(M) - \pi_{X',x}(M)| = 0.$$

Proof *Using the relation given in Equation 3.6, that is*

$$\pi_{X,x}(M) = \frac{1}{1-p'_{null}(M)}\frac{w(x)}{w'}\pi_{X',x}(M),$$

yields

$$|\pi_{X,x}(M) - \pi_{X',x}(M)| = \left|\pi_{X',x}(M)\frac{1}{1-p'_{null}(M)}\frac{w(x)}{w'} - \pi_{X',x}(M)\right|.$$

Now recall the Cauchy-Schwartz inequality. For $\mathcal{X}$ *a countable set and* $\{a(x)\}_{x\in\mathcal{X}}$ *and* $\{b(x)\}_{x\in\mathcal{X}}$ *two real-valued sequences,*

$$\sum_{x\in\mathcal{X}} a(x)b(x) \le \left\{\sum_{x\in\mathcal{X}} a^2(x)\sum_{x\in\mathcal{X}} b^2(x)\right\}^{1/2}.$$

Let $a(x) = \sqrt{\pi_{X',x}(M)}$ *and* $b(x) = \sqrt{\pi_{X',x}(M)}\frac{1}{1-p'_{null}(M)}\left|\frac{w(x)}{w'} - (1-p'_{null}(M))\right|$. *Then*

$$\begin{aligned}\sum_x a(x)b(x) &= \sum_x \pi_{X',x}(M)\frac{1}{1-p'_{null}(M)}\left|\frac{w(x)}{w'} - (1-p'_{null}(M))\right| \\ &= \sum_x \left|\pi_{X',x}(M)\frac{1}{1-p'_{null}(M)}\frac{w(x)}{w'} - \pi_{X',x}(M)\right| \\ &= \sum_x |\pi_{X,x}(M) - \pi_{X',x}(M)|\end{aligned}$$

and

$$\begin{aligned}
&\sum_x a^2(x) \sum_x b^2(x) \\
&= \sum_x \pi_{X',x}(M) \sum_x \pi_{X',x}(M) \left(\frac{1}{1 - p'_{null}(M)} \right)^2 \left(\frac{w(x)}{w'} - (1 - p'_{null}(M)) \right)^2 \\
&= \left(\frac{1}{1 - p'_{null}(M)} \right)^2 \sum_x \left(\frac{w(x)}{w'} - (1 - p'_{null}(M)) \right)^2 \pi_{X',x}(M).
\end{aligned}$$

But recalling Equation 3.5, this last sum is simply the variance of $\frac{w(X')}{w'}$ *in the primed system. Using Var to denote variance we can write*

$$\begin{aligned}
&\sum_x \left(\frac{w(x)}{w'} - (1 - p'_{null}(M)) \right)^2 \pi_{X',x}(M) \\
&= Var\left(\frac{w(X')}{w'} \right) \\
&= \frac{1}{w'^2} Var(\alpha_0 + \Delta_{CS,0} S'_0 + \ldots + \alpha_{C-1} + \Delta_{CS,C-1} S'_{C-1}) \\
&= \frac{1}{w'^2} Var(\Delta_{CS,0} S'_0 + \ldots + \Delta_{CS,C-1} S'_{C-1}) \\
&= \frac{M}{w'^2} \left[\sum_{k=0}^{C-1} \sigma'_{kk}(M) {\Delta_{CS,k}}^2 + \sum_{i \neq j} \sigma'_{ij}(M) \Delta_{CS,i} \Delta_{CS,j} \right] \\
&\to 0
\end{aligned}$$

since the scaled primed covariances are bounded and we recall that $w' = \sum_{k=0}^{C-1} \alpha_k + M\Delta_{MCS}$*. Thus, by applying the Cauchy-Schwartz inequality, we have*

$$\begin{aligned}
&\lim_{M \to \infty} \sum_x |\pi_{X,x}(M) - \pi_{X',x}(M)| \\
&\leq \lim_{M \to \infty} \sqrt{\left(\frac{1}{1 - p'_{null}(M)} \right)^2 \frac{M}{w'^2} \left[\sum_{k=0}^{C-1} \sigma'_{kk}(M) {\Delta_{CS,k}}^2 + \sum_{i \neq j} \sigma'_{ij}(M) \Delta_{CS,i} \Delta_{CS,j} \right]}
\end{aligned}$$

$$
\begin{aligned}
&= \lim_{M\to\infty}\left(\frac{1}{1-p'_{null}(M)}\right)\sqrt{\lim_{M\to\infty}\frac{M}{w'^2}\left[\sum_{k=0}^{C-1}\sigma'_{kk}(M){\Delta_{CS,k}}^2+\sum_{i\neq j}\sigma'_{ij}(M)\Delta_{CS,i}\Delta_{CS,j}\right]}\\
&= 0,
\end{aligned}
$$

which implies $\lim_{M\to\infty}\sum_x|\pi_{X,x}(M)-\pi_{X',x}(M)|=0$.

□

Next, we define

$$\overline{S}_i(M)=\frac{S_i(M)-E[S_i(M)]}{\sqrt{M}}=\frac{S_i(M)-Mq_i(M)}{\sqrt{M}}$$

and

$$\overline{S}'_i(M)=\frac{S'_i(M)-E[S'_i(M)]}{\sqrt{M}}=\frac{S'_i(M)-Mq'_i(M)}{\sqrt{M}}$$

and note that

$$\overline{S}'_i(M)=\frac{S'_i(M)-Mq_i(M)}{\sqrt{M}}+\sqrt{M}(q_i(M)-q'_i(M)).$$

By definition, each of these quantities are centred at 0. More importantly, we have introduced a specific scaling factor here. We will see that in the limit as $M\to\infty$ it is these new quantities, not the unscaled $S_i(M)$ or $S'_i(M)$, that converge to finite values. This is somewhat analogous to the infinite urn case where the weight of each colour must be centered and scaled by $\sqrt{n}$ in a particular irreducible case to acheive limiting results [34]. We note that the covariance of $\overline{S}_i(M)$ and $\overline{S}_j(M)$ is the same as the covariance of $S_i(M)/\sqrt{M}$ and $S_j(M)/\sqrt{M}$. That is, $\sigma_{ij}(M)=\mathrm{Cov}(\overline{S}_i(M),\overline{S}_j(M))$ and $\sigma'_{ij}(M)=\mathrm{Cov}(\overline{S}'_i(M),\overline{S}'_j(M))$, so we will focus on finding the limiting value of these already defined quantities.

Using Equations 3.39 and 3.40 we can now write

$$\begin{aligned}\sigma_{ij}(M) &= (E[S_i(M)\sqrt{M}S_j(M)\sqrt{M}] - E[S_i(M)\sqrt{M}]E[S_j(M)\sqrt{M}]) \\ &= (E[S_i(M)S_j(M)] - E[S_i(M)]E[S_j(M)])/M \\ &= (M-1)q_{ij}(M) + q_i(M)I_{\{i=j\}} - Mq_i(M)q_j(M) \qquad (4.1)\end{aligned}$$

and

$$\begin{aligned}\sigma'_{ij}(M) &= (E[S'_i(M)\sqrt{M}S'_j(M)\sqrt{M}] - E[S'_i(M)\sqrt{M}]E[S'_j(M)\sqrt{M}]) \\ &= (E[S'_i(M)S'_j(M)] - E[S'_i(M)]E[S'_j(M)])/M \\ &= (M-1)q'_{ij}(M) + q'_i(M)I_{\{i=j\}} - Mq'_i(M)q'_j(M). \qquad (4.2)\end{aligned}$$

Later we will focus on taking the limit of these equations, but for now, we perform some critical algebraic manipulations of Equation 4.2. We will make use of previously derived slot probability expressions for finite M given in Equations 3.20 and 3.29. Specifically, we multiply the above by Δ_{ki} and sum over i to get

$$\begin{aligned}&\sum_{i=0}^{C-1}\Delta_{ki}\sigma'_{ij}(M) \\ &= \sum_{i=0}^{C-1}\Delta_{ki}\left[(M-1)q'_{ij}(M) + q'_i(M)I_{\{i=j\}} - Mq'_i(M)q'_j(M)\right] \\ &= (M-1)\sum_{i=0}^{C-1}\Delta_{ki}q'_{ij}(M) + \sum_{i=0}^{C-1}\Delta_{ki}q'_i(M)I_{\{i=j\}} - q'_j(M)M\sum_{i=0}^{C-1}\Delta_{ki}q'_i(M) \\ &= \left(w'(1-p'_{null}(M))q_{kj}(M) - \alpha_k q'_j(M) - \Delta_{kj}q'_j(M)\right) + \Delta_{kj}q'_j(M) \\ &\quad - q'_j(M)\left(w'(1-p'_{null}(M))q_k(M) - \alpha_k\right)\end{aligned}$$

$$= \quad w'(1 - p'_{null}(M)) \left(q_{kj}(M) - q_k(M) q'_j(M) \right). \tag{4.3}$$

If we further sum over k, we get

$$\begin{aligned}
\sum_{k=0}^{C-1} \sum_{i=0}^{C-1} \Delta_{ki} \sigma'_{ij}(M) &= \sum_{i=0}^{C-1} \sum_{k=0}^{C-1} \Delta_{ki} \sigma'_{ij}(M) \\
&= \sum_{i=0}^{C-1} \Delta_{CS,i} \sigma'_{ij}(M) \\
&= \sum_{k=0}^{C-1} \left[w'(1 - p'_{null}) \left(q_{kj}(M) - q_k(M) q'_j(M) \right) \right] \\
&= w'(1 - p'_{null}) \left(q_j(M) - q'_j(M) \right). \qquad (4.4)
\end{aligned}$$

This last expression tells us something about the rate of growth of the difference $(q_i(M) - q'_i(M))$ as $M \to \infty$. Specifically, we have proved the following.

Lemma 4.2

$$\lim_{M \to \infty} \left[M \left(q_j(M) - q'_j(M) \right) \right]$$

is finite, and

$$\lim_{M \to \infty} \left[\sqrt{M} \left(q_j(M) - q'_j(M) \right) \right] = 0.$$

Proof *The result follows from the fact that the left hand side of Equation 4.4 remains bounded and the right hand side is $O(M)$.*

□

We now assume that $\overline{S}_i(M)$ and $\overline{S}'_i(M)$ converge in distribution to continuous limits. Next let $F_{ij}(M)$ and $F'_{ij}(M)$ denote the bivariate cumulative distribution functions of $\left(\overline{S}_i(M), \overline{S}_j(M) \right)$ and $\left(\overline{S}'_i(M), \overline{S}'_j(M) \right)$, respectively, and consider the following theory.

Theorem 4.3 *In the limit as $M \to \infty$ we have*

$$|F_{ij}(M) - F'_{ij}(M)| \to 0.$$

Proof *Fix $t_i, t_j \in \mathcal{R}$. Define*

$$\begin{aligned} B(t_i,t_j) &= \left\{ x : \frac{s_i(x) - Mq_i(M)}{\sqrt{M}} \le t_i, \frac{s_j(x) - Mq_j(M)}{\sqrt{M}} \le t_j \right\} \\ &= \left\{ x : \frac{s_i(x) - Mq'_i(M)}{\sqrt{M}} \le t_i - \sqrt{M}(q'_i(M) - q_i(M)), \right. \\ &\quad \left. \frac{s_j(x) - Mq'_j(M)}{\sqrt{M}} \le t_j - \sqrt{M}(q'_j(M) - q_j(M)) \right\} \end{aligned}$$

and

$$\begin{aligned} B'(t_i,t_j) &= \left\{ x : \frac{s_i(x) - Mq'_i(M)}{\sqrt{M}} \le t_i, \frac{s_j(x) - Mq'_j(M)}{\sqrt{M}} \le t_j \right\} \\ &= \left\{ x : \frac{s_i(x) - Mq_i(M)}{\sqrt{M}} \le t_i - \sqrt{M}(q_i(M) - q'_i(M)), \right. \\ &\quad \left. \frac{s_j(x) - Mq_j(M)}{\sqrt{M}} \le t_j - \sqrt{M}(q_j(M) - q'_j(M)) \right\} \end{aligned}$$

Then we have

$$\begin{aligned} &|F_{ij}(M)(t_i,t_j) - F'_{ij}(M)(t_i,t_j)| \\ &= \left| Pr\left\{ \overline{S}_i(M) \le t_i, \overline{S}_j(M) \le t_j \right\} - Pr\left\{ \overline{S}'_i(M) \le t_i, \overline{S}'_j(M) \le t_j \right\} \right| \\ &= \left| \sum_{x \in B(t_i,t_j)} \pi_{X,x}(M) - \sum_{x \in B'(t_i,t_j)} \pi_{X',x}(M) \right| \end{aligned}$$

$$
\begin{aligned}
= & \left| \sum_{x \in B(t_i,t_j) \cap B'(t_i,t_j)} \pi_{X,x}(M) + \sum_{x \in B(t_i,t_j) \cap B'(t_i,t_j)^c} \pi_{X,x}(M) \right. \\
& \left. - \sum_{x \in B(t_i,t_j) \cap B'(t_i,t_j)} \pi_{X',x}(M) - \sum_{x \in B(t_i,t_j)^c \cap B'(t_i,t_j)} \pi_{X',x}(M) \right| \\
\leq & \sum_{x \in B(t_i,t_j) \cap B'(t_i,t_j)} |\pi_{X,x}(M) - \pi_{X',x}(M)| \\
& + \left| \sum_{x \in B(t_i,t_j) \cap B'(t_i,t_j)^c} \pi_{X,x}(M) \right| + \left| \sum_{x \in B(t_i,t_j)^c \cap B'(t_i,t_j)} \pi_{X',x}(M) \right| \\
\leq & \sum_{x \in B(t_i,t_j) \cap B'(t_i,t_j)} |\pi_{X,x}(M) - \pi_{X',x}(M)| \\
& + Pr\left\{ t_i - \sqrt{M}|q_i - q_i'| < \overline{S}_i(M) \leq t_i, \overline{S}_j(M) \leq t_j \right\} \\
& + Pr\left\{ \overline{S}_i(M) \leq t_i, t_j - \sqrt{M}|q_j - q_j'| < \overline{S}_j(M) \leq t_j \right\} \\
& + Pr\left\{ t_i - \sqrt{M}|q_i - q_i'| < \overline{S}_i'(M) \leq t_i, \overline{S}_j'(M) \leq t_j \right\} \\
& + Pr\left\{ \overline{S}_i'(M) \leq t_i, t_j - \sqrt{M}|q_j - q_j'| < \overline{S}_j'(M) \leq t_j \right\} \\
\leq & \sum_{x \in B(t_i,t_j) \cap B'(t_i,t_j)} |\pi_{X,x}(M) - \pi_{X',x}(M)| \\
& + Pr\left\{ t_i - \sqrt{M}|q_i - q_i'| < \overline{S}_i(M) \leq t_i \right\} + Pr\left\{ t_j - \sqrt{M}|q_j - q_j'| < \overline{S}_j(M) \leq t_j \right\} \\
& + Pr\left\{ t_i - \sqrt{M}|q_i - q_i'| < \overline{S}_i'(M) \leq t_i \right\} + Pr\left\{ t_j - \sqrt{M}|q_j - q_j'| < \overline{S}_j'(M) \leq t_j \right\}.
\end{aligned}
$$

By Lemma 4.2 we know that $\sqrt{M}|q_i(M) - q_i'(M)| \to 0$ *and* $\sqrt{M}|q_j(M) - q_j'(M)| \to 0$ *as* $M \to \infty$. *Hence, since* $\overline{S}_i(M)$, $\overline{S}_j(M)$, $\overline{S}_i'(M)$ *and* $\overline{S}_j'(M)$ *all converge to a continuous distributions, we have*

$$Pr\left\{ t_i - \sqrt{M}|q_i - q_i'| < \overline{S}_i(M) \leq t_i \right\} \to 0,$$

$$Pr\left\{ t_j - \sqrt{M}|q_j - q_j'| < \overline{S}_j(M) \leq t_j \right\} \to 0,$$

$$Pr\left\{ t_i - \sqrt{M}|q_i - q_i'| < \overline{S}_i'(M) \leq t_i \right\} \to 0,$$

and

$$Pr\left\{t_j - \sqrt{M}|q_j - q'_j| < \overline{S}'_j(M) \leq t_j\right\} \to 0.$$

So finally, taking into account Theorem 4.1 to address the remaining term, we have shown that $|F_{ij}(M)(t_i, t_j) - F'_{ij}(M)(t_i, t_j)| \to 0$ *as* $M \to \infty$ *as required.*

□

Corollary 4.4 *In the limit, slot probabilities, scaled means and scaled covariances in the unprimed and primed systems are equal. Specifically, we have*

$$\lim_{M\to\infty} q_i(M) = \lim_{M\to\infty} q'_i(M),$$

$$\lim_{M\to\infty} q_{ij}(M) = \lim_{M\to\infty} q'_{ij}(M)$$

and

$$\lim_{M\to\infty} \sigma_{ij}(M) = \lim_{M\to\infty} \sigma'_{ij}(M).$$

Proof *Each of the slot probabilities, scaled means and scaled covariances in the unprimed and primed systems can be thought of as functions of* $F_{ij}(M)$ *and* $F'_{ij}(M)$ *respectively. Since* $|F_{ij}(M) - F'_{ij}(M)| \to 0$*, each of these limiting quantities converge to the same values.*

□

So we see that in the limit, many steady state unprimed and primed quantities of interest are the same. We will use this key fact several times in the remaining sections of this chapter.

4.2 Limiting Slot Probabilities

In Section 3.3 we introduced slot probabilities $q_i(M)$ and $q_{ij}(M)$ that turned out to be at the heart of the first and second order moment calculations. In this section we investigate these probabilities as $M \to \infty$ assuming they converge. We will let $q_i = \lim_{M\to\infty} q_i(M)$ and $q_{ij} = \lim_{M\to\infty} q_{ij}(M)$. Also, we take $q'_i = \lim_{M\to\infty} q'_i(M)$ and $q'_{ij} = \lim_{M\to\infty} q'_{ij}(M)$. More generally, and for the remainder of the document, quantities which do not specify the dependence on M explicitly will refer to their respective limiting value as $M \to \infty$.

The first step is to take the limit of Equation 3.20. The left hand side is straightforward. We have

$$\lim_{M\to\infty}\left[\left(\frac{\left(\sum_{k=0}^{C-1}\alpha_k\right)}{M}+\Delta_{MCS}\right)(1-p'_{null}(M))q_i(M)\right]=\Delta_{MCS}(1-p'_{null})q_i.$$

Similarly, the right hand side is

$$\lim_{M\to\infty}\left[\frac{\alpha_i}{M}+\sum_{j=0}^{C-1}\Delta_{ij}r'_j(M)q_j(M)\right]=\sum_{j=0}^{C-1}\Delta_{ij}\left(\lim_{M\to\infty}\left(r'_j(M)q_j(M)\right)\right)=\sum_{j=0}^{C-1}\Delta_{ij}q_j,$$

does not present a challenge once it is realized that

$$\lim_{M\to\infty}\left(r'_j(M)q_i(M)\right)=\lim_{M\to\infty}q'_j(M)=q_j$$

owing to Corollary 4.4.

Remark 4.5 *Alternatively, we can avoid the use of Corollary 4.4 here if we can address* $\lim_{M\to\infty} r'_j(M)$. *This limit is essentially a statement about the relative growth rates of* $p'_{null|j}(M)$ *and* $p'_{null}(M)$. *In fact we can also handle this limit directly using*

Corollary A.2. One advantage of using this alternate logic is that Corollary 4.4 relies on the assumption that σ'_{ij} is bounded for all $0 \leq i, j \leq C-1$, whereas Corollary A.2 has no such restriction. In reality the additional assumption is not much of a restraint since we intend to go to calculate σ_{ij} and σ'_{ij}, and at that point we will assume that these objects exist anyway.

Continuing, we now have

$$\sum_{j=0}^{C-1} \Delta_{ij} q_j = \Delta_{MCS}(1 - p'_{null}) q_i,$$

which can be written more succinctly as

$$\Delta q = \lambda q, \tag{4.5}$$

using λ to take the place of $\Delta_{MCS}(1-p'_{null})$ and q to represent the vector of q_i values. It is clear that q is a right eigenvector of Δ with eigenvalue λ. We just need a way to identify this eigenvalue, eigenvector pair. The following lemma provides a method.

Lemma 4.6 *$\lambda = \Delta_{MCS}(1 - p'_{null})$ is the largest, real, strictly positive eigenvalue (Perron-Frobenius) of Δ, and q is the unique (after normalization) associated real, strictly positive eigenvector.*

Proof *Recall that we have assumed our weight matrix is irreducible and non-negative. Moreover, by the definition of q, as a vector of probabilities, we have $q \geq 0$. The lemma then follows from the Subinvariance Theorem for irreducible non-negative matrices as given in [41], Theorem 1.6, p. 23.*

□

At this point we have a method to calculate q for any urn in the set $\mathcal{L}$. Still, we are interested in the forms of the special cases introduced in Section 3.1.

The first case is simple. When the columns of Δ are constant, Equation 4.5 reduces to

$$\Delta q = \Delta_{CCS} q \tag{4.6}$$

since $\Delta_{MCS} = \Delta_{CCS}$ and $p'_{null} = 0$ in this case. The same theory applies for finding q, namely, it is the eigenvector associated with the largest real eigenvalue, which in this case is simply Δ_{CCS}.

Next we turn our attention to those urns in the intersection with $\mathcal{D}$. We have previously seen that the expression for $q_i(M)$ was in a different form in this case, one that involved only a single $q_i(M)$. Hence, this time we can simply take the limit and immediately get q_i without having to solve an eigen problem. Here we get

$$\begin{aligned} q_i &= \lim_{M\to\infty} q_i(M) \\ &= \lim_{M\to\infty} \frac{\left(\frac{\alpha_i}{M} + \delta_i\right)}{\left(\left(\frac{\left(\sum_{k=0}^{C-1}\alpha_k\right)}{M} + \Delta_{MCS}\right)(1 - p'_{null}(M)) - \tau_i r'_i(M)\right)} \\ &= \frac{\delta_i}{(\Delta_{MCS}(1 - p'_{null}) - \tau_i)} \\ &= \frac{\delta_i}{\lambda - \tau_i}. \end{aligned} \tag{4.7}$$

Next, if we further have that the column sums of Δ are constant, ie. in the set $\mathcal{L} \cap \mathcal{D} \cap \mathcal{C}$, then we have

$$\begin{aligned} q_i &= \lim_{M\to\infty} q_i(M) \\ &= \lim_{M\to\infty} \frac{\left(\frac{\alpha_i}{M} + \delta_i\right)}{\left(\frac{\left(\sum_{k=0}^{C-1}\alpha_k\right)}{M} + \sum_{k=0}^{C-1}\delta_k\right)} \\ &= \frac{\delta_i}{\sum_{k=0}^{C-1}\delta_k}. \end{aligned} \tag{4.8}$$

For the limiting second order slot probability, q_{ij}, we begin with Equation 3.29. Taking the limit and applying Corollary 4.4 we arrive at

$$\begin{aligned} &\lim_{M\to\infty}\left[\left(\frac{\left(\sum_{k=0}^{C-1}\alpha_k\right)}{M} + \Delta_{MCS}\right)(1 - p'_{null}(M))q_{ij}(M)\right] \\ &= \Delta_{MCS}(1 - p'_{null})q_{ij} \\ &= \lim_{M\to\infty}\left[\frac{\alpha_i r'_j(M)q_j(M)}{M} + \frac{(M-1)}{M}\sum_{k=0}^{C-1}\Delta_{ik}r'_{kj}(M)q_{kj}(M) + \frac{\Delta_{ij}r'_j(M)q_j(M)}{M}\right] \\ &= \sum_{k=0}^{C-1}\Delta_{ik}q_{kj}. \end{aligned}$$

Remark 4.7 *As we noted during the first order slot probability calculation, Corollary A.2 can be used in place of Corollary 4.4 here if we desire. As we mentioned earlier, however, there is little reason not to use the more comprehensive Corollary 4.4 at this time given our ultimate goals.*

In matrix-vector notation, this time using $q^{(j)}$ as the vector $(q_{0j}, q_{1j}, \ldots, q_{C-1,j})^T$, we can write the familiar equation

$$\Delta q^{(j)} = \lambda q^{(j)}. \tag{4.9}$$

On the surface, this seems no different than Equation 4.5. Indeed, $q^{(j)}$ must be the same right eigenvector of Δ, up to normalization, with eigenvalue λ. That is, we know $q^{(j)}$ and q are proportional. But the sum of the components of $q^{(j)}$ is q_j, so the proportionality constant is q_j. Hence, we have shown that

$$q^{(j)} = q_j q$$

in general, which implies that

$$q_{ij} = q_i q_j \tag{4.10}$$

for all i, j. Thus, in the limit the contents of consecutive slots are independent.

Next, we can consider the intersections of $\mathcal{C}$ and $\mathcal{D}$ with $\mathcal{L}$. Not too much is to be gained this time, other than confirming that the special cases also readily imply that the limiting second order slot probability is the product of the two corresponding limiting first order slot probabilities.

In the constant weight, but otherwise general linear functional weight components, case we simply get

$$\Delta q^{(j)} = \Delta_{CCS} q^{(j)}, \tag{4.11}$$

from which we do not glean any new insight.

When the urn belongs to $\mathcal{D}$, and otherwise is general, we can infer

$$q_{ij} = \lim_{M\to\infty} \left[\frac{\left(\frac{\alpha_i}{M} + \delta_i + \frac{\tau_i I_{\{i=j\}}}{M}\right) r'_j(M) q_j(M)}{\left(\left(\frac{\left(\sum_{k=0}^{C-1} \alpha_k\right)}{M} + \Delta_{MCS}\right)(1 - p'_{null}(M)) - \frac{(M-1)}{M}\tau_i r'_{ij}(M)\right)} \right]$$

$$
\begin{aligned}
&= \frac{\delta_i}{(\Delta_{MCS}(1-p'_{null})-\tau_i)}q_j \\
&= q_i q_j. \qquad (4.12)
\end{aligned}
$$

This highlights very explicitly how q_{ij} becomes $q_i q_j$ in the limit.

Finally, if the urn is both constant weight and has definite functional weight components, we can calculate

$$
\begin{aligned}
q_{ij} &= \lim_{M\to\infty}\left[\frac{\left(\frac{\alpha_i}{M}+\delta_i+\frac{\left(\Delta_{CCS}-\sum_{k=0}^{C-1}\delta_k\right)I_{\{i=j\}}}{M}\right)q_j(M)}{\left(\frac{\left(\sum_{k=0}^{C-1}\alpha_k\right)}{M}+\frac{\Delta_{CCS}}{M}+\frac{(M-1)}{M}\sum_{k=0}^{C-1}\delta_k\right)}\right] \\
&= \frac{\delta_i}{\sum_{k=0}^{C-1}\delta_k}q_j \\
&= q_i q_j. \qquad (4.13)
\end{aligned}
$$

As always, this form is the simplest we present. This simplicity, however, comes at the cost of the most restrictions on our urn.

4.3 Limiting Scaled Mean and Covariance

At this point we have derived a variety of basic measures for our functional weight finite urn. For constant memory M we have considered slot probabilities and then later used these to deduce mean and covariance expressions. In the previous section we extended our look at slot probabilities to include limiting results as $M \to \infty$. The purpose of this section is to leverage our work with the limiting slot probabilities and derive limiting mean and covariance results, assuming these quantities exist.

We must exercise caution when when taking limits of complex expressions. Looking back at the finite mean and covariance results of Section 3.4 we immediately realize that without proper scaling, these quantities are at risk of diverging as $M \to \infty$. For example, since we already accept that $\lim_{M\to\infty} q_i(M)$ and $\lim_{M\to\infty} q_i'(M)$ converge, we must certainly believe that $E[S_i(M)] = Mq_i(M) \to \infty$ and $E[S_i'(M)] = Mq_i'(M) \to \infty$. Intuitively this makes perfect sense. The expected number of slots, out of a total of M, containing colour i should grow without bound as M increases provided the urn parameters remain unchanged. Similarly, judging from Equations 3.39 and 3.40, it is not clear whether $(E[S_i(M)S_j(M)] - E[S_i(M)]E[S_j(M)])$ or $(E[S_i'(M)S_j'(M)] - E[S_i'(M)]E[S_j'(M)])$ converge either. Indeed, we will see that we must first scale the previous finite mean and covariance expressions in order to achieve meaningful limiting results here.

Given our previous work with limiting slot probabilities, it is not hard to guess how to scale the mean $E[S_i(M)]$ and $E[S_i'(M)]$. In fact, we have essentially already seen the results before. They are presented here mainly for completeness.

Lemma 4.8 *The limiting scaled mean vectors are given by*

$$\lim_{M\to\infty} [E[S(M)]/M] = q$$

and

$$\lim_{M\to\infty} [E[W(M)]/M] = \lambda q.$$

Proof *The first expression is trivial given our previous work. On the individual component level, we immediately have*

$$\lim_{M\to\infty}[E[S_i(M)]/M] = \lim_{M\to\infty}\left[\frac{1}{M}Mq_i(M)\right] = \lim_{M\to\infty} q_i(M) = q_i.$$

The second expression is also easy to develop. Consider that

$$\lim_{M\to\infty}[E[W_i(M)]/M] = \lim_{M\to\infty}\left[\frac{1}{M}M\sum_{j=0}^{C-1}\Delta_{ij}q_j(M)\right] = \sum_{j=0}^{C-1}\Delta_{ij}q_j.$$

But since

$$\Delta q = \lambda q,$$

this implies $(\lim_{M\to\infty}[E[W_i(M)]/M])_i = \lambda q$ *as required.*

□

Remark 4.9 *It is important to note that the mean urn composition above is the same as that presented by Janson in [34] for time limiting scaled infinite memory urns.*

We now consider the limiting scaled covariances, $\sigma_{ij} = \lim_{M\to\infty}\sigma_{ij}(M)$ and $\sigma'_{ij} = \lim_{M\to\infty}\sigma'_{ij}(M)$. Once again, a critical factor of M seems to hold the key to convergence. The quantities $(E[S_i(M)S_j(M)]-E[S_i(M)]E[S_j(M)])$ and $(E[S'_i(M)S'_j(M)]-E[S'_i(M)]E[S'_j(M)])$ grow too quickly in the limit, however, we will see that the scaled values $\sigma_{ij}(M)$ and $\sigma'_{ij}(M)$ can be found as $M\to\infty$.

Recall that by Equations 4.1 and 4.2 we have the relationships

$$\begin{aligned}
\sigma_{ij}(M) &= (E[S_i(M)S_j(M)] - E[S_i(M)]E[S_j(M)])/M \\
&= (M-1)q_{ij}(M) + q_i(M)I_{\{i=j\}} - Mq_i(M)q_j(M) \\
&= q_i(M)I_{\{i=j\}} - q_{ij}(M) + M(q_{ij}(M) - q_i(M)q_j(M))
\end{aligned}$$

and

$$\begin{aligned}\sigma'_{ij}(M) &= (E[S'_i(M)S'_j(M)] - E[S'_i(M)]E[S'_j(M)])/M \\ &= (M-1)q'_{ij}(M) + q'_i(M)I_{\{i=j\}} - Mq'_i(M)q'_j(M) \\ &= q'_i(M)I_{\{i=j\}} - q'_{ij}(M) + M(q'_{ij}(M) - q'_i(M)q'_j(M)).\end{aligned}$$

At this point though, we pause to reflect on the challenge before us. The above expressions contain exactly three terms each on the right hand side. Taking the limit as $M \to \infty$ would not present any difficulties if it were not for the respective third terms. Despite showing earlier that $\lim_{M\to\infty} q_{ij}(M) = \lim_{M\to\infty}(q_i(M)q_j(M))$ and $\lim_{M\to\infty} q'_{ij}(M) = \lim_{M\to\infty}(q'_i(M)q'_j(M))$, or equivalently, that $\lim_{M\to\infty}(q_{ij}(M) - q_i(M)q_j(M)) = 0$ and $\lim_{M\to\infty}(q'_{ij}(M) - q'_i(M)q'_j(M)) = 0$, we do not have information about the rate at which either of these latter expressions hold true. Hence, at this point we cannot say much about either $\lim_{M\to\infty} M(q_{ij}(M) - q_i(M)q_j(M))$ or $\lim_{M\to\infty} M(q'_{ij}(M) - q'_i(M)q'_j(M))$. Luckily, the method we present to solve for σ_{ij} and σ'_{ij} does not rely on finding these limits explicitly.

Our procedure resumes calculations using the unprimed expression for $\sigma_{ij}(M)$ as a launching point. It makes use of the previously of developed Equations 4.3 and 4.4. We compute

$$\begin{aligned}\sigma_{ij}(M) &= q_i(M)I_{\{i=j\}} - q_{ij}(M) + M(q_{ij}(M) - q_i(M)q_j(M)) \\ &= q_i(M)I_{\{i=j\}} - q_{ij}(M) + M(q_{ij}(M) - q_i(M)q'_j(M) + q_i(M)q'_j(M) \\ &\quad - q_i(M)q_j(M))\end{aligned}$$

$$
\begin{aligned}
&= q_i(M)I_{\{i=j\}} - q_{ij}(M) + M(q_{ij}(M) - q_i(M)q'_j(M)) - q_i(M)M(q_j(M) \\
&\quad - q'_j(M)) \\
&= q_i(M)I_{\{i=j\}} - q_{ij}(M) + \frac{M}{w'(1-p'_{null}(M))}\sum_{k=0}^{C-1}\Delta_{ik}\sigma'_{kj}(M) \\
&\quad - q_i(M)\frac{M}{w'(1-p'_{null}(M))}\sum_{k=0}^{C-1}\Delta_{CS,k}\sigma'_{kj}(M) \\
&= q_i(M)I_{\{i=j\}} - q_{ij}(M) \\
&\quad + \frac{M}{w'(1-p'_{null}(M))}\sum_{k=0}^{C-1}\left(\Delta_{ik} - q_i(M)\Delta_{CS,k}\right)\sigma'_{kj}(M) \\
&= q_i(M)I_{\{i=j\}} - q_{ij}(M) \\
&\quad + \frac{1}{\frac{w'}{M}(1-p'_{null}(M))}\sum_{k=0}^{C-1}\left(\Delta_{ik} - q_i(M)\Delta_{CS,k}\right)\sigma'_{kj}(M). \qquad (4.14)
\end{aligned}
$$

At this stage we once again have a single equation with both unprimed and primed quantities. As we have done before, the way to continue lies in knowing that $\sigma'_{ij}(M)$ and $\sigma_{ij}(M)$ are indistinguishable in the limit, a fact previously understood from Corollary 4.4.

Remark 4.10 *We cannot avoid use of Corollary 4.4 at this point. Logically then, we must adhere to the assumption that σ_{ij} is bounded for all $0 \leq i, j \leq C-1$. This fact is assured, however, by our implicit assumption that σ_{ij} exist for all $0 \leq i, j \leq C-1$ in this section.*

Thus, taking the limit of Equation 4.14 gives

$$
\begin{aligned}
\sigma_{ij} &= \lim_{M\to\infty}\sigma_{ij}(M) \\
&= \lim_{M\to\infty}\Big[q_i(M)I_{\{i=j\}} - q_{ij}(M)
\end{aligned}
$$

$$+ \frac{1}{\frac{w'}{M}(1 - p'_{null}(M))} \sum_{k=0}^{C-1} (\Delta_{ik} - q_i(M)\Delta_{CS,k})\, \sigma'_{kj}(M) \Big]$$
$$= \quad q_i I_{\{i=j\}} - q_i q_j + \frac{1}{\lambda} \sum_{k=0}^{C-1} (\Delta_{ik} - q_i \Delta_{CS,k})\, \sigma_{kj}.$$

Remark 4.11 *Essentially, our careful algebraic manipulations to this point have allowed us to manage the term $M(q_{ij}(M) - q_i(M)q_j(M))$ in the limit, provided we know that $\sigma_{ij} = \sigma'_{ij}$. Perhaps more insight into this sufficient condition is given in Theorem A.3. There it is shown more explicitly that the validity of the statement $\sigma_{ij} = \sigma'_{ij}$ has ramifications for limits of the form M times a difference, where the difference itself is known to vanish as $M \to \infty$.*

Moving on, suppose Σ_{IID} is a matrix such that

$$\Sigma_{IID} = \Big(q_i I_{\{i=j\}} - q_i q_j \Big)_{ij}.$$

Σ_{IID} is the covariance matrix of a system in which the probability of drawing a given colour is always according to a Multinomial(M, q) distribution, independent of the state of the urn.

Remark 4.12 *The rank of Σ_{IID} is $C - 1$. To see this fact, we could show that the vector q is not a linear combination of the first $C - 1$ column vectors of Σ_{IID}. We can do this by noting that starting with the assumption that q is a linear combination of the first $C - 1$ column vectors leads to a contradiction, or possibly by showing that the matrix Σ_{IID}, with the last column replaced by q, has determinant $q_0 q_1 \cdots q_{C-1}$, which must be strictly positive. Either way, the rank of Σ_{IID} is at least $C - 1$. Yet, certainly Σ_{IID} cannot have full rank C since the columns sum to 0. So we determine the rank to be exactly $C - 1$.*

At this stage we can switch to matrix-vector notation and write

$$\Sigma = \Sigma_{IID} + \frac{1}{\lambda}(\Delta - q\Delta_{CS}{}^t)\Sigma,$$

or

$$\left[I - \frac{1}{\lambda}(\Delta - q\Delta_{CS}{}^t)\right]\Sigma = \Sigma_{IID},$$

where $\Sigma = (\sigma_{ij})_{ij}$ is our desired covariance matrix and we let

$$\Delta_{CS} = (\Delta_{CS,0}, \Delta_{CS,1}, \ldots, \Delta_{CS,C-1})^t$$

be the vector of column sums of Δ, and $\Delta_{CS}{}^t$ its vector transpose. We take I to be the identity matrix as usual.

Pushing forward we define

$$L = \frac{1}{\lambda}(\Delta - q\Delta_{CS}{}^t)$$

and note that we can solve for Σ provided we can always invert $[I - L]$. The following theorem ensures that this is always possible.

Theorem 4.13 *The limiting scaled covariance matrices can be expressed as*

$$\left(\lim_{M\to\infty}[(E[S_i(M)S_j(M)] - E[S_i(M)]E[S_j(M)])/M]\right)_{ij} = \Sigma = [I - L]^{-1}\Sigma_{IID}$$

and

$$\left(\lim_{M\to\infty}[(E[W_i(M)W_j(M)] - E[W_i(M)]E[W_j(M)])/M]\right)_{ij} = \Delta\Sigma\Delta^t,$$

where Δ^t is the matrix transpose of Δ.

Proof *Given all of our previous work, it suffices to show that* $[I - L]^{-1}$ *exists for the proof of the first statement. Initially we will focus on determining this fact.*

L *is a square, finite,* $C \times C$ *matrix with real components. General matrix theory tells us that* $[I - L]^{-1}$ *exists, and more specifically,* $[I - L]^{-1} = \sum_{k=0}^{\infty} L^k$*, where* $L^0 = I$ *by definition, provided* $L^k \to 0$ *elementwise as* $k \to \infty$ *[41].*

Now, consider that

$$\begin{aligned}
(\Delta - q\Delta_{CS}{}^t)(\Delta - q\Delta_{CS}{}^t) &= (\Delta - q\Delta_{CS}{}^t)\Delta - (\Delta - q\Delta_{CS}{}^t)q\Delta_{CS}{}^t \\
&= (\Delta - q\Delta_{CS}{}^t)\Delta - \Delta q\Delta_{CS}{}^t + q\Delta_{CS}{}^t q\Delta_{CS}{}^t \\
&= (\Delta - q\Delta_{CS}{}^t)\Delta - \lambda q\Delta_{CS}{}^t + q\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}\Delta q\Delta_{CS}{}^t \\
&= (\Delta - q\Delta_{CS}{}^t)\Delta - \lambda q\Delta_{CS}{}^t + q\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}\lambda q\Delta_{CS}{}^t \\
&= (\Delta - q\Delta_{CS}{}^t)\Delta - \lambda q\Delta_{CS}{}^t + \lambda q\Delta_{CS}{}^t \\
&= (\Delta - q\Delta_{CS}{}^t)\Delta
\end{aligned}$$

so we may write

$$\begin{aligned}
L^k &= \left[\frac{1}{\lambda}(\Delta - q\Delta_{CS}{}^t)\right]^k = \frac{1}{\lambda^k}(\Delta - q\Delta_{CS}{}^t)^k \\
&= \frac{1}{\lambda}(\Delta - q\Delta_{CS}{}^t)\frac{1}{\lambda^{k-1}}\Delta^{k-1}.
\end{aligned}$$

Next, recall that λ *is the largest real eigenvalue of the irreducible non-negative weight matrix* Δ*. Corresponding to* λ *is the right eigenvector* q*. These facts were ascertained earlier in Lemma 4.6. Perron-Frobenius theory further states that there exists*

a strictly positive left eigenvector, say v, associated with λ [41]. Moreover, it is also discussed in [41] that

$$\lim_{k\to\infty} \frac{1}{\lambda^{k-1}}\Delta^{k-1} = qv^t.$$

Altogether then we have

$$\begin{aligned}
\lim_{k\to\infty} L^k &= \lim_{k\to\infty}\left[\frac{1}{\lambda}(\Delta - q\Delta_{CS}{}^t)\frac{1}{(\lambda)^{k-1}}\Delta^{k-1}\right] \\
&= \frac{1}{\lambda}(\Delta - q\Delta_{CS}{}^t)\lim_{k\to\infty}\left[\frac{1}{(\lambda)^{k-1}}\Delta^{k-1}\right] \\
&= \frac{1}{\lambda}(\Delta - q\Delta_{CS}{}^t)qv^t \\
&= \frac{1}{\lambda}(\Delta q - q\Delta_{CS}{}^t q)v^t \\
&= \frac{1}{\lambda}(\lambda q - q\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}\Delta q)v^t \\
&= \frac{1}{\lambda}(\lambda q - q\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}\lambda q)v^t \\
&= (q-q)v^t \\
&= 0
\end{aligned}$$

as required.

The second statement is really a consequence of the first. We have

$$\begin{aligned}
&\lim_{M\to\infty}\left[(E[W_i(M)W_j(M)] - E[W_i(M)]E[W_j(M)])/M\right] \\
&= \lim_{M\to\infty}\left[\frac{1}{M}\left(E\left[\left(\sum_{k=0}^{C-1}\Delta_{ik}S_k(M)\right)\left(\sum_{l=0}^{C-1}\Delta_{jl}S_l(M)\right)\right]\right.\right. \\
&\quad \left.\left. - E\left[\left(\sum_{k=0}^{C-1}\Delta_{ik}S_k(M)\right)\right]E\left[\left(\sum_{l=0}^{C-1}\Delta_{jl}S_l(M)\right)\right]\right)\right]
\end{aligned}$$

$$
\begin{aligned}
&= \lim_{M\to\infty}\left[\frac{1}{M}\left(\sum_{k=0}^{C-1}\sum_{l=0}^{C-1}\Delta_{ik}\Delta_{jl}E[S_k(M)S_l(M)]\right.\right.\\
&\quad\left.\left.-\sum_{k=0}^{C-1}\sum_{l=0}^{C-1}\Delta_{ik}\Delta_{jl}E[S_k(M)]E[S_l(M)]\right)\right]\\
&= \sum_{k=0}^{C-1}\sum_{l=0}^{C-1}\Delta_{ik}\Delta_{jl}\lim_{M\to\infty}[(E[S_k(M)S_l(M)]-E[S_k(M)]E[S_l(M)])/M]\\
&= \sum_{k=0}^{C-1}\sum_{l=0}^{C-1}\Delta_{ik}\sigma_{kl}\Delta_{jl}.
\end{aligned}
$$

In matrix notation the last expression is $\Delta\Sigma\Delta^t$.

□

Remark 4.14 *From the proof of Theorem 4.13 we see that the representation*

$$[I-L]^{-1}=\sum_{k=0}^{\infty}L^k$$

exists as well. With this in mind, we also have another representation for Σ. *If we desire, we have*

$$
\begin{aligned}
\Sigma &= \sum_{k=0}^{\infty}L^k\Sigma_{IID}\\
&= \left[I+\sum_{k=1}^{\infty}L^k\right]\Sigma_{IID}\\
&= \left[I+\sum_{k=1}^{\infty}\frac{1}{\lambda}(\Delta-q\Delta_{CS}{}^t)\frac{1}{\lambda^{k-1}}\Delta^{k-1}\right]\Sigma_{IID}\\
&= \left[I+\frac{1}{\lambda}(\Delta-q\Delta_{CS}{}^t)\sum_{k=0}^{\infty}\frac{1}{\lambda^k}\Delta^k\right]\Sigma_{IID},
\end{aligned}
$$

where we take $\Delta^0=I$ *to be consistent with the previous convention of* $L^0=I$.

Remark 4.15 *The rank of Σ is $C-1$. This follows from the fact that Σ_{IID} is the product of $[I-L]$ and Σ. From linear algebra, the rank of Σ_{IID} must be at most the minimum of the rank of $[I-L]$ and the rank of Σ. Since $[I-L]$ is invertible, it has full rank, and so the rank of Σ_{IID} is actually at most the rank of Σ. If the rank of Σ was anything less than $C-1$, it would contradict the fact that Σ_{IID} has rank exactly $C-1$.*

Remark 4.16 *Most importantly, notice that unlike [34], here we only find a single expression for the covariance matrix of the weights of each colour type. That is, we have considered the entire irreducible class of finite urns at once using a single scaling factor of $\sqrt{M}$.*

With this in mind, it is not hard to show that our covariance expressions above are not necessarily the same as any of the the time limiting scaled covariances derived in [34], even in the case where the magnitude of the second largest eigenvalue is less than 1/2 the magnitude of the largest eigenvalue. An example that highlights this fact is given later in Section 4.4.2.

As always, we may consider the special sub cases of the set $\mathcal{L}$ separately. The usual approach is to back up to some point in the derivation of the general expression and begin again with the further restrictions dictated by the sub case under consideration. Typically, this leads to expressions that are simpler in some sense, for a minimal amount of additional work. This time, however, we will see that this is not always the case.

We begin, as always, with urns in the set $\mathcal{L} \cap \mathcal{C}$. Following the same method, and making all obvious simplifications, can lead to

$$\left[I - \frac{1}{\Delta_{CCS}}\Delta\right]\Sigma = \Sigma_{IID}. \tag{4.15}$$

While correct, we cannot go any farther toward isolating Σ. Here, $\left[I - \frac{1}{\Delta_{CCS}}\Delta\right]$ is singular since it has an eigenvalue equal to 0. Without an inverse, it seems we have to settle for using the general linear form expression in this case.

Next we consider definite form urns. Typically we discover more explicit and straightforward expressions in these cases. Judging from equation 4.14 it should be feasible here too since we have explicit expressions for $q_i(M)$ and $q_{ij}(M)$. Indeed, it is straightforward to sub in these components and taking the limit, invoking Corollary 4.4 as necessary. The calculations are tedious, and very similar to other derivations thus far, so we omit much of the algebra here. In the end we arrive at the expression

$$\sigma_{ij} = \left(\frac{\lambda}{\lambda - \tau_i}\right)\left(q_i I_{\{i=j\}} - q_i q_j\right) - \sum_{k=0}^{C-1} \frac{q_i \Delta_{CS,k}}{\lambda - \tau_i}\sigma_{kj}.$$

If we define a diagonal matrix

$$U_1 = \left(\frac{\lambda}{\lambda - \tau_i}\right)_{ij},$$

and in a similar way, another matrix

$$U_2 = \left(\frac{q_i}{\lambda - \tau_i}\Delta_{CS,k}\right)_{ik},$$

then we can write

$$\Sigma = U_1\Sigma_{IID} - U_2\Sigma$$

or

$$[I + U_2]\Sigma = U_1\Sigma_{IID}.$$

As in the more general case above, in order to solve for Σ we must show that the inverse of $[I + U_2]$ exists.

Lemma 4.17 *In the linear definite case we also have*

$$\left(\lim_{M\to\infty}[(E[S_i(M)S_j(M)] - E[S_i(M)]E[S_j(M)])/M]\right)_{ij} = \Sigma = [I + U_2]^{-1}U_1\Sigma_{IID}.$$

Proof *We need only show that the matrix $[I + U_2]$ has an inverse. Our approach relies on the fact that $[I + U_2]^{-1}$ exists if and only if $[I + U_2]$ has no 0 eigenvalues. We use this method because the specific form of U_2 allows us to easily guess and verify all C eigenvalue and eigenvector pairs of $[I + U_2]$.*

First, consider the $C - 1$ vectors

$$(\Delta_{CS,1}, -\Delta_{CS,0}, 0, \ldots, 0), (0, \Delta_{CS,2}, -\Delta_{CS,1}, \ldots, 0), \ldots, (0, \ldots, \Delta_{CS,C-1}, -\Delta_{CS,C-2}).$$

It is easy to check that these form an independent set of eigenvectors of U_2, each with eigenvalue 0.

Second, we acknowledge that U_2 contains strictly positive elements. So by Perron-Frobenius theory for primitive matrices, there must also exist another strictly positive eigenvalue with some corresponding eigenvector.

Hence, the complete set of eigenvalues of $[I + U_2]$, which we get by adding 1 to the set of eigenvalues of U_2, must consist of $C - 1$ 1's and another of the form $1 + \epsilon$, with

$\epsilon > 0$. *Since we have determined that none of the eigenvalues of* $[I + U_2]$ *is 0,* $[I + U_2]$ *is invertible.*

□

It is not clear that this form is more desirable in any sense, compared with the general form result of earlier. Still, it is an interesting alternative. More importantly perhaps, regardless of which form is used to construct Σ in this case, we can simplify the limiting scaled weight covariance matrix $\Delta\Sigma\Delta^t$. We will see that the alternative form comes as a result of the increased structure of Δ.

For the moment, observe that in the set $\mathcal{L} \cap \mathcal{D}$ we can write the weight matrix, given in Equation 3.10, as $\Delta = T + D$ where we take

$$T = \begin{bmatrix} \tau_0 & 0 & \cdots & 0 \\ 0 & \tau_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \tau_{C-1} \end{bmatrix}$$

and

$$D = \begin{bmatrix} \delta_0 & \delta_0 & \cdots & \delta_0 \\ \delta_1 & \delta_1 & \cdots & \delta_1 \\ \vdots & \vdots & \ddots & \vdots \\ \delta_{C-1} & \delta_{C-1} & \cdots & \delta_{C-1} \end{bmatrix}.$$

We will use this new representation of Δ to state the following lemma. Note that T is diagonal and D has repeated columns.

Lemma 4.18 *In the linear definite case we also have*

$$\left(\lim_{M\to\infty}\left[(E[W_i(M)W_j(M)] - E[W_i(M)]E[W_j(M)])/M\right]\right)_{ij} = T\Sigma T.$$

Proof *The key here is the fact that our covariance matrix has rows and columns that sum to 0. Hence, if we multiply our covariance matrix on the left by a matrix with repeated columns, the product is 0. Similarly, if we multiply our covariance matrix on the right by a matrix with repeated rows, the product is again 0. With this in mind we simply calculate*

$$\Delta\Sigma\Delta^t = [T+D]\Sigma[T+D]^t = T\Sigma[T+D]^t = T\Sigma[T+D^t] = T\Sigma T.$$

□

Finally, we also consider urns with both definite form functional weights and constant column sum weight matrices. Once again beginning with Equation 4.14 we can directly calculate

$$\sigma_{ij} = \frac{\Delta_{CCS}}{\sum_{k=0}^{C-1}\delta_k}\left(q_i I_{\{i=j\}} - q_i q_j\right).$$

This allows us to state a final result in the following lemma.

Lemma 4.19 *In the linear definite and constant weight case we also have*

$$\left(\lim_{M\to\infty}\left[(E[S_i(M)S_j(M)] - E[S_i(M)]E[S_j(M)])/M\right]\right)_{ij} = \Sigma = \frac{\Delta_{CCS}}{\sum_{k=0}^{C-1}\delta_k}\Sigma_{IID}.$$

Proof *Nothing further is required. The result is simply a matrix form of our previous work.*

□

This expression for Σ is much simpler than the previous ones. Moreover, we see next that we can further reduce the expression for the limiting scaled weight covariance matrix as well.

Lemma 4.20 *In the linear definite and constant weight case we also have*

$$\left(\lim_{M\to\infty}[(E[W_i(M)W_j(M)] - E[W_i(M)]E[W_j(M)])/M]\right)_{ij} = \frac{\tau^2\Delta_{CCS}}{\sum_{k=0}^{C-1}\delta_k}\Sigma_{IID}.$$

Proof *Since we are in the linear definite case we can already write*

$$\left(\lim_{M\to\infty}[(E[W_i(M)W_j(M)] - E[W_i(M)]E[W_j(M)])/M]\right)_{ij} = T\Sigma T.$$

But since we also assume constant weight here, we further know that $T = \tau I$ thanks to Equation 3.14. Thus we must have

$$\left(\lim_{M\to\infty}[(E[W_i(M)W_j(M)] - E[W_i(M)]E[W_j(M)])/M]\right)_{ij} = \tau I\Sigma\tau I = \tau^2\Sigma$$

where we take Σ for this case as given in Lemma 4.20.

□

Each covariance expression above has determined Σ to be the product of some multiplying matrix, call it A, and Σ_{IID}. There is an interesting observation to be made about the various multiplying matrices. They are not direct algebraic reductions of one another, even when the constraints on the urn clearly dictate they belong in the same class. A simple numeric example confirms this fact. Take for instance the definite and constant weight urn characterized by the weight matrix

$$\Delta = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}.$$

Using the general form expression we calculate

$$[I - L]^{-1} = \begin{bmatrix} 1.222222 & -0.111111 & -0.111111 \\ -0.111111 & 1.222222 & -0.111111 \\ -0.111111 & -0.111111 & 1.222222 \end{bmatrix},$$

while using the definite, but not constant weight, form expression yields

$$[I + U_2]^{-1}U_1 = \begin{bmatrix} 1.079365 & -0.253968 & -0.253968 \\ -0.253968 & 1.079365 & -0.253968 \\ -0.253968 & -0.253968 & 1.079365 \end{bmatrix}.$$

Finally, the definite and constant weight form result gives

$$\frac{\Delta_{MCS}}{\sum_{k=0}^{C-1} \delta_k} I = \begin{bmatrix} 1.333333 & 0 & 0 \\ 0 & 1.333333 & 0 \\ 0 & 0 & 1.333333 \end{bmatrix},$$

which is different from the previous two yet again. Of course the final product, Σ, must be equal in each case since each of Theorem 4.13, Lemma 4.17 and Lemma 4.19 are valid. Still this example highlights that the multiplying matrix A is not unique, and hence, it is worth investigating further.

The critical observation, which is also exploited in the proof of Lemma 4.18, is that

$$[I - L + R]\Sigma = [I - L]\Sigma$$

is true for any matrix R with repeated columns. This allows us to characterize a class of sufficient multiplying matrices that may be used to calculate Σ in the form

$$\Sigma = A^{-1}\Sigma_{IID}.$$

In fact, at this point we can restate Theorem 4.13, generalizing it to include this latest observation.

Theorem 4.21 *Suppose R is a matrix with repeated columns. Provided $[I - L + R]$ is invertible, the limiting scaled covariance matrices can be expressed as*

$$\left(\lim_{M\to\infty}[(E[S_i(M)S_j(M)] - E[S_i(M)]E[S_j(M)])/M]\right)_{ij} = \Sigma = [I - L + R]^{-1}\Sigma_{IID}$$

and

$$\left(\lim_{M\to\infty}[(E[W_i(M)W_j(M)] - E[W_i(M)]E[W_j(M)])/M]\right)_{ij} = \Delta\Sigma\Delta^t,$$

where Δ^t is the matrix transpose of Δ.

Moreover, if $R = 0$ then $[I - L]$ is always invertible.

Proof *Concerning the second statement, suppose initially that R is the zero matrix. This case is handled by Theorem 4.13. Next, consider $R \neq 0$ with repeated columns. The rest follows from the observation that $[I - L + R]\Sigma = [I - L]\Sigma = \Sigma_{IID}$ for such a matrix R and the fact it is assumed that $[I - L + R]$ is invertible.*

The proof of the form of the limiting scaled urn composition covariance matrix in the second statement remains unchanged from that given in Theorem 4.13.

□

Remark 4.22 *This form of the theorem still ensures that we can always find* Σ*, and hence* $\Delta\Sigma\Delta^t$ *as well, by taking* $R = 0$*. Yet, depending on the constraints imposed on the urn in question, it states that there may be other forms available. It allows for the possibility that an alternate form yields an expression for* Σ *that is more desirable.*

In the next section we illustrate the versatility of Theorem 4.21. We now use it to rederive the special case expressions for Σ presented in Lemma 4.17 and Lemma 4.19.

4.4 Examples

4.4.1 Rederiving Special Case Results From General Expressions

Previously we have seen both general expressions, which hold anytime our urn is in the set $\mathcal{L}$, and a couple of special case results, for limiting means and covariances. The special case expressions were not always simple reductions of the corresponding general form. Usually, they were derived separately. This first example points out that we can achieve the special case results directly from the general forms.

Consider our calculation of the limiting mean vector. We know that the mean vector is the right eigenvector of Δ associated with the largest real eigenvalue. But suppose

that we are in the definite form class. Then $\Delta = T + D$, as defined before, and we have

$$\begin{aligned}[T + D]q &= (\tau_0 q_0, \tau_1 q_1, \ldots, \tau_{C-1} q_{C-1})^t + (\delta_0, \delta_1, \ldots, \delta_{C-1})^t \\ &= \lambda q \\ &= (\lambda q_0, \lambda q_1, \ldots, \lambda q_{C-1})^t,\end{aligned}$$

which implies that

$$\tau_i q_i + \delta_i = \lambda q_i$$

or

$$q_i = \frac{\delta_i}{\lambda - \tau_i}$$

exactly as before. Moreover, if we are also in the set $\mathcal{C}$, we likewise get

$$q_i = \frac{\delta_i}{\lambda - \tau} = \frac{\delta_i}{\sum_{k=0}^{C-1} \delta k}.$$

Next, consider the definite case once more and this time imagine calculating Σ using Theorem 4.21. Let

$$R = \frac{1}{\lambda} D,$$

and check that

$$\begin{aligned}
[I - L + R] &= \left(I - \frac{1}{\lambda}(\Delta - q\Delta_{CS}{}^t) + R\right) \\
&= \left(I - \frac{1}{\lambda}T + \frac{1}{\lambda}q\Delta_{CS}{}^t\right) \\
&= \left(\frac{1}{\lambda}(\lambda I - T) + \frac{1}{\lambda}q\Delta_{CS}{}^t\right) \\
&= U_1{}^{-1} + U_1{}^{-1}U_2 = U_1{}^{-1}[I + U_2].
\end{aligned}$$

But note that the inverse of this last matrix is $[I + U_2]^{-1}U_1$, which appeared as the multiplying matrix in the definite result already obtained by an alternate method.

We demonstrate a similar trick for the set $\mathcal{L} \cap \mathcal{D} \cap \mathcal{C}$. This time take

$$R = \frac{1}{\Delta_{CCS}}\left(D - q\Delta_{CS}{}^t\right).$$

Note that R still has repeated columns since the components of Δ_{CS} are all Δ_{CCS} in the constant weight case. Then

$$\begin{aligned}
[I - L + R] &= \left(I - \frac{1}{\Delta_{CCS}}(\Delta - q\Delta_{CS}{}^t) + R\right) \\
&= \left(I - \frac{1}{\Delta_{CCS}}T\right) \\
&= \frac{1}{\Delta_{CCS}}\left(\Delta_{CCS}I - T\right) \\
&= \frac{1}{\Delta_{CCS}}(\Delta_{CCS} - \tau)I \\
&= \frac{\sum_{k=0}^{C-1}\delta_k}{\Delta_{CCS}}I.
\end{aligned}$$

This is exactly the A that would correspond to the required $A^{-1} = \frac{\Delta_{CCS}}{\sum_{k=0}^{C-1}\delta_k}I$ we have seen before in Lemma 4.19. Here it is quite clear that in view of Theorem 4.21 an

appropriate choice of R can significantly simplify the general result available using only Theorem 4.13.

So we see that the alternate, and sometimes more explicit, forms of q and Σ can be derived directly from the general forms. With these we can construct all special case first and second order slot probabilities, means and covariances.

4.4.2 2×2 Weight Matrix

As an example of calculating some of the limiting quantities outlined in this chapter, let us consider a 2 colour linear functional weight finite urn in detail. The case $C = 2$ is interesting in that it is small enough that the calculations can be done explicitly. Moreover, this case always belongs in the set $\mathcal{D}$ as well, as we can clearly see by its characterizing weight matrix

$$\Delta = \begin{bmatrix} \Delta_{00} & \Delta_{01} \\ \Delta_{10} & \Delta_{11} \end{bmatrix} = \begin{bmatrix} \Delta_{00} & \delta_0 \\ \delta_1 & \Delta_{11} \end{bmatrix}.$$

The first thing to calculate is the set of eigenvalues of Δ. It is easy to check that they are

$$\frac{1}{2}\left(\Delta_{00} + \Delta_{11} \pm \sqrt{{\Delta_{00}}^2 - 2\Delta_{00}\Delta_{11} + {\Delta_{11}}^2 + 4\Delta_{01}\Delta_{10}}\right).$$

Using the largest eigenvalue we see that

$$\lambda = \Delta_{MCS}(1 - p'_{null}) = \frac{1}{2}\left(\Delta_{00} + \Delta_{11} + \sqrt{{\Delta_{00}}^2 - 2\Delta_{00}\Delta_{11} + {\Delta_{11}}^2 + 4\Delta_{01}\Delta_{10}}\right).$$

Note that from this we can deduce

$$p'_{null} = \frac{2\Delta_{MCS} - \Delta_{00} - \Delta_{11} - \sqrt{{\Delta_{00}}^2 - 2\Delta_{00}\Delta_{11} + {\Delta_{11}}^2 + 4\Delta_{01}\Delta_{10}}}{2\Delta_{MCS}}.$$

Next, we can compute the eigenvector corresponding to the largest eigenvalue, λ. After normalization and a bit of algebra we can obtain

$$\begin{aligned} q_0 &= \frac{2\Delta_{01}}{\Delta_{11} - \Delta_{00} + \sqrt{{\Delta_{00}}^2 - 2\Delta_{00}\Delta_{11} + {\Delta_{11}}^2 + 4\Delta_{01}\Delta_{10}} + 2\Delta_{01}} \\ &= \frac{\Delta_{01}}{\frac{1}{2}\left(\Delta_{00} + \Delta_{11} + \sqrt{{\Delta_{00}}^2 - 2\Delta_{00}\Delta_{11} + {\Delta_{11}}^2 + 4\Delta_{01}\Delta_{10}}\right) - (\Delta_{00} - \Delta_{01})} \end{aligned}$$

and

$$\begin{aligned} q_1 &= \frac{2\Delta_{10}}{\Delta_{00} - \Delta_{11} + \sqrt{{\Delta_{00}}^2 - 2\Delta_{00}\Delta_{11} + {\Delta_{11}}^2 + 4\Delta_{01}\Delta_{10}} + 2\Delta_{10}} \\ &= \frac{\Delta_{10}}{\frac{1}{2}\left(\Delta_{00} + \Delta_{11} + \sqrt{{\Delta_{00}}^2 - 2\Delta_{00}\Delta_{11} + {\Delta_{11}}^2 + 4\Delta_{01}\Delta_{10}}\right) - (\Delta_{11} - \Delta_{10})}, \end{aligned}$$

where the second equality is shown for each component so that we may recognize the explicit form we found earlier in the definite case theory.

At this point we note that the expression for λ and these basic components of q are identical to those given in Example 7.2 of [34]. This confirms a remark earlier that our limiting scaled functional weight finite urn mean composition is the same as for a time limiting scaled infinite memory urn.

Recall that finding $q = (q_0, q_1)$, as we have done, actually tells us a lot more than just the first order slot probabilities. We know that q is the limiting scaled mean of

the number of each colour in the urn, and λq tells us the limiting mean of the scaled urn composition. Finally, if we desire, we can also construct the second order slot probabilities from the first order components since $q_{ij} = q_i q_j$.

If we know that the column sums of the weight matrix are equal, we can exploit this fact to simplify q further. In this case, where specifically we have

$$\Delta_{CCS} = \Delta_{00} + \Delta_{10} = \Delta_{01} + \Delta_{11},$$

we know the largest eigenvalue of Δ should be Δ_{CCS} and we must have $p'_{null} = 0$. We can double check this by verifying that

$$\sqrt{{\Delta_{00}}^2 - 2\Delta_{00}\Delta_{11} + {\Delta_{11}}^2 + 4\Delta_{01}\Delta_{10}} = \Delta_{01} + \Delta_{10},$$

so that ultimately the above eigenvalues from the general form become

$$\frac{1}{2}\left(\Delta_{00} + \Delta_{11} \pm (\Delta_{01} + \Delta_{10})\right),$$

from which we infer that

$$\lambda = \frac{1}{2}\left(\Delta_{00} + \Delta_{01} + \Delta_{10} + \Delta_{11}\right) = \frac{1}{2} 2\Delta_{CCS} = \Delta_{CCS}.$$

Moreover, we find that the expressions above for q_0 and q_1 readily reduce to

$$q_0 = \frac{\Delta_{01}}{\Delta_{01} + \Delta_{10}}$$

and

$$q_1 = \frac{\Delta_{10}}{\Delta_{01} + \Delta_{10}}$$

as expected based on our theory for this case.

Next, we focus our attention on the limiting covariance matrix for our $C = 2$ example. We know that any expression for Σ involves the matrix Σ_{IID}, so we begin by calculating

$$\Sigma_{IID} = \begin{bmatrix} q_0 - q_0 q_0 & -q_0 q_1 \\ -q_1 q_0 & q_1 - q_1 q_1 \end{bmatrix} = \begin{bmatrix} q_0 q_1 & -q_0 q_1 \\ -q_0 q_1 & q_0 q_1 \end{bmatrix} = q_0 q_1 \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

With this in hand, let us tackle the general form with $R = 0$. That is, we need to calculate $[I - L]^{-1}$. After some tedious algebra, which we omit here to save space, we get

$$[I - L]^{-1} = \frac{1}{(\lambda - \tau_0 q_1 - \tau_1 q_0)} \begin{bmatrix} \lambda - \Delta_{11} + q_1 \Delta_{CS,1} & \Delta_{01} - q_0 \Delta_{CS,1} \\ \Delta_{10} - q_1 \Delta_{CS,0} & \lambda - \Delta_{00} + q_0 \Delta_{CS,0} \end{bmatrix}.$$

Thus

$$\begin{aligned}
\Sigma &= [I - L]^{-1} \Sigma_{IID} \\
&= \frac{1}{(\lambda - \tau_0 q_1 - \tau_1 q_0)} \begin{bmatrix} \lambda - \Delta_{11} + q_1 \Delta_{CS,1} & \Delta_{01} - q_0 \Delta_{CS,1} \\ \Delta_{10} - q_1 \Delta_{CS,0} & \lambda - \Delta_{00} + q_0 \Delta_{CS,0} \end{bmatrix} q_0 q_1 \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \\
&= \frac{q_0 q_1}{(\lambda - \tau_0 q_1 - \tau_1 q_0)} \begin{bmatrix} \lambda - \Delta_{11} + q_1 \Delta_{CS,1} & \Delta_{01} - q_0 \Delta_{CS,1} \\ \Delta_{10} - q_1 \Delta_{CS,0} & \lambda - \Delta_{00} + q_0 \Delta_{CS,0} \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \\
&= \frac{q_0 q_1}{(\lambda - \tau_0 q_1 - \tau_1 q_0)} \begin{bmatrix} \lambda - \Delta_{CS,1} + \Delta_{CS,1} & -(\lambda - \Delta_{CS,1} + \Delta_{CS,1}) \\ -(\lambda - \Delta_{CS,0} + \Delta_{CS,0}) & \lambda - \Delta_{CS,0} + \Delta_{CS,0} \end{bmatrix} \\
&= \frac{\lambda q_0 q_1}{(\lambda - \tau_0 q_1 - \tau_1 q_0)} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.
\end{aligned}$$

We know that Σ is the limiting covariance matrix of the scaled numbers of each colour type in the urn. If we want the limiting covariance matrix of the scaled weights of each type, we further calculate

$$\begin{aligned}
\Delta\Sigma\Delta^t &= \begin{bmatrix} \Delta_{00} & \Delta_{01} \\ \Delta_{10} & \Delta_{11} \end{bmatrix} \frac{\lambda q_0 q_1}{(\lambda - \tau_0 q_1 - \tau_1 q_0)} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \Delta_{00} & \Delta_{10} \\ \Delta_{01} & \Delta_{11} \end{bmatrix} \\
&= \frac{\lambda q_0 q_1}{(\lambda - \tau_0 q_1 - \tau_1 q_0)} \begin{bmatrix} \tau_0 & -\tau_0 \\ -\tau_1 & \tau_1 \end{bmatrix} \begin{bmatrix} \Delta_{00} & \Delta_{10} \\ \Delta_{01} & \Delta_{11} \end{bmatrix} \\
&= \frac{\lambda q_0 q_1}{(\lambda - \tau_0 q_1 - \tau_1 q_0)} \begin{bmatrix} {\tau_0}^2 & -\tau_0\tau_1 \\ -\tau_0\tau_1 & {\tau_1}^2 \end{bmatrix}.
\end{aligned}$$

Before we move on to some special cases, we compare this last limiting covariance matrix with the expressions presented in Example 7.2 of [34]. We recall that the results for a limiting scaled infinite memory urn in [34] actually break into various cases depending on a relationship between the largest eigenvalue of Δ, in our notation λ, and the next largest eigenvalue. When $C = 2$, there is only one other eigenvalue to consider, and the cases come about depending on whether this other eigenvalue is less than, equal to or greater than an amount $\lambda/2$. Based on the general expressions for the eigenvalues above, in our notation this translates into whether the sum $\Delta_{00} + \Delta_{11}$ is less than, equal to or greater than $3\sqrt{{\Delta_{00}}^2 - 2\Delta_{00}\Delta_{11} + {\Delta_{11}}^2 + 4\Delta_{01}\Delta_{10}}$. It must be stressed, however, that the result for our finite urn does not exhibit this case dependence.

In actual fact, we will only compare to the two out of three cases where there are explicit expressions for the limiting scaled covariance matrix in [34]. The following are Janson's results translated into our notation. If we have

$$\Delta_{00} + \Delta_{11} < 3\sqrt{{\Delta_{00}}^2 - 2\Delta_{00}\Delta_{11} + {\Delta_{11}}^2 + 4\Delta_{01}\Delta_{10}}$$

then a time scaling of $\sqrt{n}$ leads to a limiting scaled covariance matrix given by

$$\Sigma_T = \frac{\lambda q_0 q_1}{\left(\frac{3}{2}\sqrt{{\Delta_{00}}^2 - 2\Delta_{00}\Delta_{11} + {\Delta_{11}}^2 + 4\Delta_{01}\Delta_{10}} - \frac{1}{2}(\Delta_{00} + \Delta_{11})\right)} \begin{bmatrix} {\tau_0}^2 & -\tau_0\tau_1 \\ -\tau_0\tau_1 & {\tau_1}^2 \end{bmatrix},$$

while if

$$\Delta_{00} + \Delta_{11} = 3\sqrt{{\Delta_{00}}^2 - 2\Delta_{00}\Delta_{11} + {\Delta_{11}}^2 + 4\Delta_{01}\Delta_{10}}$$

then a time scaling of $\sqrt{n \ln n}$ is used and the matrix is given by

$$\Sigma_T = q_0 q_1 \begin{bmatrix} {\tau_0}^2 & -\tau_0\tau_1 \\ -\tau_0\tau_1 & {\tau_1}^2 \end{bmatrix}.$$

A concrete example shows that both these expressions differ from our $\Delta\Sigma\Delta^t$. For the first case suppose we take

$$\Delta = \begin{bmatrix} 7 & 1 \\ 5 & 3 \end{bmatrix},$$

a weight matrix we considered at the beginning of this chapter. Then

$$\Delta\Sigma\Delta^t = \frac{1}{3}\begin{bmatrix} {\tau_0}^2 & -\tau_0\tau_1 \\ -\tau_0\tau_1 & {\tau_1}^2 \end{bmatrix} = \begin{bmatrix} 12 & 4 \\ 4 & 4/3 \end{bmatrix},$$

whereas

$$\Sigma_T = \frac{1}{2}\begin{bmatrix} {\tau_0}^2 & -\tau_0\tau_1 \\ -\tau_0\tau_1 & {\tau_1}^2 \end{bmatrix} = \begin{bmatrix} 18 & 6 \\ 6 & 2 \end{bmatrix},$$

as we have seen before. Note that the covariance entries are different. It is now clear why the exact finite distribution, with large M, is not well approximated by the limiting normal distribution plotted in Figure 4.1. Interestingly however, an excellent approximation is provided by a limiting normal with the proper variance taken from $\Delta\Sigma\Delta^t$ above. We see this fact in Figure 4.2 where we have once again plotted the distribution of $(W_0(10000) - E[W_0(10000)])/\sqrt{10000}$, taken from $\pi(10000)$ found using $C = 2$, $\alpha = (1, 1)$ and Δ as above, this time compared against a normal distribution with mean 0 and variance 12.

In this case the curves in the plot are essentially coincident. This not only validates our calculation of the covariance matrix, but more importantly perhaps, it implies that the limiting distribution of a sequence of functional weight finite urn stationary distributions is normal. We will come back to this observation later in Section 4.5.

Similarly, for the second case suppose we have

$$\Delta = \begin{bmatrix} 11 & 1 \\ 5 & 7 \end{bmatrix}.$$

Then

$$\Delta\Sigma\Delta^t = \frac{1}{2}\begin{bmatrix} {\tau_0}^2 & -\tau_0\tau_1 \\ -\tau_0\tau_1 & {\tau_1}^2 \end{bmatrix} = \begin{bmatrix} 50 & -10 \\ -10 & 2 \end{bmatrix},$$

whereas

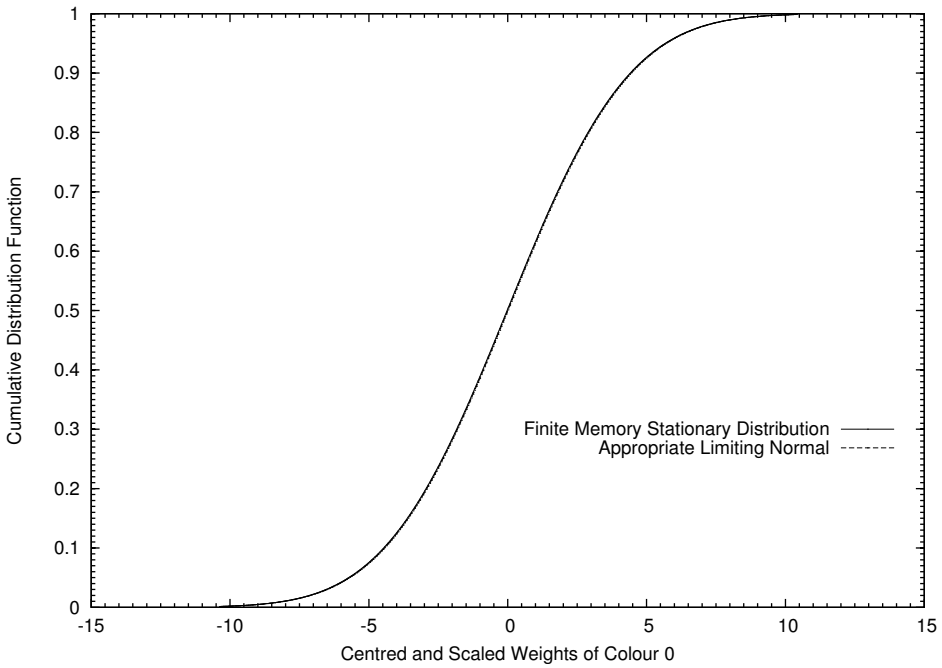

Figure 4.2: Comparison of the Cumulative Distribution Function of the weight of colour 0, as calculated from the stationary distribution with $M = 10000$, and the limiting normal distribution with an appropriate variance.

$$\Sigma_T = \frac{1}{4}\begin{bmatrix} {\tau_0}^2 & -\tau_0\tau_1 \\ -\tau_0\tau_1 & {\tau_1}^2 \end{bmatrix} = \begin{bmatrix} 25 & -5 \\ -5 & 1 \end{bmatrix}.$$

Getting back to illustrating our theory, we see that since a 2×2 weight matrix must always belong to the set $\mathcal{L} \cap \mathcal{D}$, it is interesting to calculate $[I + U_2]^{-1}U_1\Sigma_{IID}$ and verify that it is equal to the previous expression we deduced for Σ. After some additional calculating we can produce

$$[I + U_2]^{-1}U_1 = \frac{\lambda}{\theta}\begin{bmatrix} \lambda - \tau_1 + q_1\Delta_{CS,1} & -q_0\Delta_{CS,1} \\ -q_1\Delta_{CS,0} & \lambda - \tau_0 + q_0\Delta_{CS,0} \end{bmatrix},$$

where $\theta = ((\lambda-\tau_0)(\lambda-\tau_1)+q_0\Delta_{CS,0}(\lambda-\tau_1)+q_1\Delta_{CS,1}(\lambda-\tau_0))$. Recall that there is no reason for this multiplying matrix to be the same as $[I - L]^{-1}$. However, continuing with the definite expression we find

$$\begin{aligned}
\Sigma &= [I + U_2]^{-1}U_1\Sigma_{IID} \\
&= \frac{\lambda}{\theta}\begin{bmatrix} \lambda - \tau_1 + q_1\Delta_{CS,1} & -q_0\Delta_{CS,1} \\ -q_1\Delta_{CS,0} & \lambda - \tau_0 + q_0\Delta_{CS,0} \end{bmatrix} q_0q_1\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \\
&= \frac{\lambda q_0 q_1}{\theta}\begin{bmatrix} \lambda - \tau_1 + q_1\Delta_{CS,1} + q_0\Delta_{CS,1} & -(\lambda - \tau_1 + q_1\Delta_{CS,1} + q_0\Delta_{CS,1}) \\ -(\lambda - \tau_0 + q_0\Delta_{CS,0} + q_1\Delta_{CS,0}) & \lambda - \tau_0 + q_0\Delta_{CS,0} + q_1\Delta_{CS,0} \end{bmatrix} \\
&= \frac{\lambda q_0 q_1}{\theta}\begin{bmatrix} \lambda - \tau_1 + \Delta_{CS,1} & -(\lambda - \tau_1 + \Delta_{CS,1}) \\ -(\lambda - \tau_0 + \Delta_{CS,0}) & \lambda - \tau_0 + \Delta_{CS,0} \end{bmatrix} \\
&= \frac{\lambda q_0 q_1(\lambda + \Delta_{01} + \Delta_{10})}{\theta}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix},
\end{aligned}$$

which we know must be the same as before. In fact, it is straightforward to verify that $\theta = (\lambda+\Delta_{01}+\Delta_{10})(\lambda-\tau_0 q_1-\tau_1 q_0)$ and conclude that this expression is identical to the one derived from the general theory.

Finally, we look at Σ in a constant weight scenario. Realizing that

$$\tau = \tau_0 = \tau_1 = \Delta_{CCS} - (\Delta_{01} + \Delta_{10})$$

in this context, implies

$$\begin{aligned} \frac{\lambda}{(\lambda - q_0\tau_1 - q_1\tau_0)} &= \frac{\Delta_{CCS}}{(\Delta_{CCS} - q_0(\Delta_{CCS} - (\Delta_{01} + \Delta_{10})) - q_1(\Delta_{CCS} - (\Delta_{01} + \Delta_{10}))} \\ &= \frac{\Delta_{CCS}}{(\Delta_{01} + \Delta_{10})}, \end{aligned}$$

our general expression reduces to

$$\Sigma = \frac{\Delta_{CCS}}{(\Delta_{01} + \Delta_{10})} q_0 q_1 \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \frac{\Delta_{CCS} q_0 q_1}{(\Delta_{01} + \Delta_{10})} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

We can readily confirm that this is the same as that predicted by our constant weight theory. The limiting scaled covariance of the weights of the urn is similarly reduced to

$$\begin{aligned} \Delta\Sigma\Delta^t &= \begin{bmatrix} \Delta_{00} & \Delta_{01} \\ \Delta_{10} & \Delta_{11} \end{bmatrix} \frac{\Delta_{CCS} q_0 q_1}{(\Delta_{01} + \Delta_{10})} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \Delta_{00} & \Delta_{10} \\ \Delta_{01} & \Delta_{11} \end{bmatrix} \\ &= \frac{\Delta_{CCS} q_0 q_1}{(\Delta_{01} + \Delta_{10})} \begin{bmatrix} \tau_0 & -\tau_0 \\ -\tau_1 & \tau_1 \end{bmatrix} \begin{bmatrix} \Delta_{00} & \Delta_{10} \\ \Delta_{01} & \Delta_{11} \end{bmatrix} \end{aligned}$$

$$
\begin{aligned}
&= \frac{\Delta_{CCS} q_0 q_1}{(\Delta_{01} + \Delta_{10})} \begin{bmatrix} {\tau_0}^2 & -\tau_0\tau_1 \\ -\tau_0\tau_1 & {\tau_1}^2 \end{bmatrix} \\
&= \frac{\Delta_{CCS} q_0 q_1 \tau^2}{(\Delta_{01} + \Delta_{10})} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.
\end{aligned}
$$

Even in this more restricted setting, however, the above expression is different than the corresponding result for a time limiting infinite urn. Next we present Janson's reduced expressions in the special constant weight case. If

$$\Delta_{00} + \Delta_{11} < 3(\Delta_{01} + \Delta_{10})$$

we get

$$\Sigma_T = \frac{\Delta_{CCS} q_0 q_1 \tau^2}{((\Delta_{01} + \Delta_{10}) - \tau)} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix},$$

while if

$$\Delta_{00} + \Delta_{11} = 3(\Delta_{01} + \Delta_{10})$$

we have

$$\Sigma_T = q_0 q_1 \tau^2 \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

But once again both expressions differ from our result when the column sums are constant. Suppose for instance we take

$$\Delta = \begin{bmatrix} 3 & 1 \\ 5 & 7 \end{bmatrix}$$

to satisfy the conditions of Janson's first case. Then

$$\Delta\Sigma\Delta^t = \frac{5}{27}\tau^2 \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 20/27 & -20/27 \\ -20/27 & 20/27 \end{bmatrix},$$

whereas

$$\Sigma_T = \frac{5}{18}\tau^2 \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 10/9 & -10/9 \\ -10/9 & 10/9 \end{bmatrix}.$$

Similarly, to illustrate the second case suppose we have

$$\Delta = \begin{bmatrix} 7 & 1 \\ 5 & 11 \end{bmatrix}.$$

Then

$$\Delta\Sigma\Delta^t = \frac{5}{18}\tau^2 \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 10 & -10 \\ -10 & 10 \end{bmatrix},$$

whereas

$$\Sigma_T = \frac{5}{36}\tau^2 \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 5 & -5 \\ -5 & 5 \end{bmatrix}.$$

It should be clear by the end of this example that the limiting distribution of our centred and scaled process is different, strictly speaking, from that of classic infinite urns. On the other hand, we have seen that the mean vectors for both the limiting finite and infinite urn processes are identical, and even the covariance matrices take a similar form. In the next section we use this information to formally conjecture that the distribution of our limiting scaled finite urn process is in the same family as a limiting scaled infinite urn process, but with different parameters.

4.5 Limiting Normality Conjecture

For this section, consider an urn in the set $\mathcal{L}$ according to the classification given in Figure 3.1. Recall that corresponding to each member of this class of urns is a characterizing weight matrix, Δ. The coefficients of Δ are used in linear weight functions of the form $g_i(x) = \sum_{j=0}^{C-1} \Delta_{ij} s_j(x)$. We further assume, as usual, that Δ is irreducible and non-negative.

At this point, we have thoroughly investigated the limiting scaled mean and covariance calculations for this class. We are reminded that our derivations necessarily assume that these objects exist. Numerically, this fact is not in question, but technically, the assumption must remain.

The main goal of this section though, is to propose the forms of limiting sequences of various scaled random vectors. The components of these vectors are familiar by now. Recall the centred and scaled processes $\overline{S}_i(M)$ and $\overline{S}'_i(M)$ as previously defined. The means of these processes are always zero by definition. Also, we know that the limiting covariance matrix of a multivariate distribution of these processes, if it exists, must be given by Σ. Moreover, we let $\overline{W}_i(M) = \sum_{j=0}^{C-1} \Delta_{ij} \overline{S}_j(M)$ and $\overline{W}'_i(M) = \sum_{j=0}^{C-1} \Delta_{ij} \overline{S}'_j(M)$ to be the centred and scaled weight processes for unprimed and primed urns respectively. These also have zero means. This time however, we know that the limiting covariance matrix of a multivariate distribution of these scaled weight processes, provided it exists, must be $\Delta\Sigma\Delta^t$.

With these definitions firmly in mind, we put forth the following conjecture regarding the forms of specific limiting distributions for finite urns.

Conjecture 4.23 *For finite urns with memory M in the class $\mathcal{L}$, in the limit as $M \to \infty$, the quantities q, q', Σ and Σ' exist, and we have*

$$\Big(\overline{S}_0(M), \overline{S}_1(M), \ldots, \overline{S}_{C-1}(M)\Big) \to \mathit{Normal}(0, \Sigma)$$

and

$$\Big(\overline{S}'_0(M), \overline{S}'_1(M), \ldots, \overline{S}'_{C-1}(M)\Big) \to \mathit{Normal}(0, \Sigma).$$

Moreover, it is true that

$$\Big(\overline{W}_0(M), \overline{W}_1(M), \ldots, \overline{W}_{C-1}(M)\Big) \to \mathit{Normal}(0, \Delta\Sigma\Delta^t)$$

and

$$\Big(\overline{W}'_0(M), \overline{W}'_1(M), \ldots, \overline{W}'_{C-1}(M)\Big) \to \mathit{Normal}(0, \Delta\Sigma\Delta^t).$$

Remark 4.24 *The components $\overline{W}_i(M)$ and $\overline{W}'_i(M)$ are linear combinations of $\overline{S}_j(M)$ for $0 \leq j \leq C-1$ and $\overline{S}'_j(M)$ for $0 \leq j \leq C-1$ respectively. If it can be shown that Σ exists and the first and second statements can be justified, then the third and fourth statements follow immediately owing to the fact that a linear combination of normal distributions is also normally distributed. Therefore, it is sufficient to focus on the first pair of statements.*

Remark 4.25 *Notice also that by Theorem 4.3 we can already say that $\overline{S}_i(M)$ and $\overline{S}'_i(M)$ approach the same limit. Hence, the second statement is equivalent to the first. Furthermore, by the previous remark, $\overline{W}_i(M)$ and $\overline{W}'_i(M)$ approach the same*

limit. Hence, the fourth statement is equivalent to the third as well. We present both expressions in either pair to be explicit and complete in terms of the limiting forms for each of the multivariate processes we have introduced in this work.

The above predictions are based on a number of factors. First, there are strong similarities between the evolution of infinite urns and finite urns. Since it is well known that the former tends to a limiting normal distribution in several key irreducible cases [34], it is reasonable to suspect a limiting normal to appear in the context of the latter as well. Second, we have already remarked that the elements of our fundamental stationary urn process, $X(M)$, are exchangeable. We note that many central limit theorems exist for sequences of exchangeable random variables. Third, using an appropriate covariance matrix, we can produce strong empirical evidence for the limiting normality hypothesis in the case of a sequence of stationary finite urns. We can refer to Figure 4.2, for instance, and the discussion surrounding it. That result is just a single example that is indicative of a large number of other weight matrices we have considered numerically.

Before concluding this section, we consider another way to view our finite urn, one that may aid thinking about the above conjecture in the future. The discussion is valid in the special case of $\mathcal{L} \cap \mathcal{D} \cap \mathcal{C}$.

The main observation is that we can view aspects of the evolution of our stationary finite urn as a sequence of independent Bernoulli trials, where the critical Bernoulli parameter has a specific Beta distribution. This alternate way of looking at urn processes is well known in the context of infinite urns. It has been used extensively ever since their introduction [36]. Here, it potentially allows us to say more about the existence of the quantities q, q', Σ and Σ'. To get a feel for how this new representation

applies to finite urns, we consider the simple 2 colour case initially.

Suppose that p_0, the probability of drawing colour 0 in an unprimed system, has a Beta distribution with parameters $\beta_0 = \frac{\alpha_0 + M\delta_0}{\tau}$ and $\beta_1 = \frac{\alpha_1 + M\delta_1}{\tau}$ such that we can view draws as independent Bernoulli trials with parameter p_0. Indeed, we can confirm that this procedure is an equivalent way of thinking of a finite urn by simply calculating the probability of s_0 colour 0 slots emerging out of any M draws. In the new setting this probability becomes

$$\begin{aligned}
&\binom{M}{s_0} \int_0^1 {p_0}^{s_0}(1-p_0)^{M-s_0} \frac{\Gamma(\beta_0+\beta_1)}{\Gamma(\beta_0)\Gamma(\beta_1)} {p_0}^{\beta_0-1}(1-p_0)^{\beta_1-1} dp_0 \\
= \quad &\binom{M}{s_0} \frac{\Gamma(\beta_0+\beta_1)}{\Gamma(\beta_0)\Gamma(\beta_1)} \int_0^1 {p_0}^{\beta_0+s_0-1}(1-p_0)^{\beta_1+M-s_0-1} dp_0 \\
= \quad &\binom{M}{s_0} \frac{\Gamma(\beta_0+\beta_1)}{\Gamma(\beta_0)\Gamma(\beta_1)} \frac{\Gamma(\beta_0+s_0)\Gamma(\beta_1+M-s_0)}{\Gamma(\beta_0+\beta_1+M)}
\end{aligned}$$

where we take

$$\Gamma(\beta) = \int_0^\infty x^{\beta-1} e^{-x} dx$$

to be the standard gamma function. Next, owing to the well known fact that $\Gamma(\beta) = (\beta-1)\Gamma(\beta-1)$ for $\beta > 1$, we can reduce this expression further to

$$\begin{aligned}
&\binom{M}{s_0} \frac{\Gamma(\beta_0+\beta_1)}{\Gamma(\beta_0)\Gamma(\beta_1)} \frac{\Gamma(\beta_0+s_0)\Gamma(\beta_1+M-s_0)}{\Gamma(\beta_0+\beta_1+M)} \\
= \quad &\binom{M}{s_0} \frac{\Gamma(\beta_0+\beta_1)}{\Gamma(\beta_0)\Gamma(\beta_1)} \frac{\left(\prod_{l_0=0}^{s_0-1}(\beta_0+l_0)\right)\Gamma(\beta_0)\left(\prod_{l_1=0}^{M-s_0-1}(\beta_1+l_1)\right)\Gamma(\beta_1)}{\left(\prod_{l=0}^{M-1}(\beta_0+\beta_1+l)\right)\Gamma(\beta_0+\beta_1)} \\
= \quad &\binom{M}{s_0} \frac{\left(\prod_{l_0=0}^{s_0-1}(\beta_0+l_0)\right)\left(\prod_{l_1=0}^{M-s_0-1}(\beta_1+l_1)\right)}{\left(\prod_{l=0}^{M-1}(\beta_0+\beta_1+l)\right)}
\end{aligned}$$

$$= \binom{M}{s_0} \frac{\prod_{l_0=0}^{s_0-1} (\alpha_0 + M\delta_0 + l_0\tau) \prod_{l_1=0}^{M-s_0-1} (\alpha_1 + M\delta_1 + l_1\tau)}{\prod_{l=0}^{M-1} (\alpha_0 + M\delta_0 + \alpha_1 + M\delta_1 + l\tau)}.$$

But now it is easy to compare this expression to the stationary form presented in Section 3.2 and deduce that the two are identical. Hence, we can conclude that thinking of successive draws from our stationary finite urn as Bernoulli trials with parameter $p_0 \sim \text{Beta}(\beta_0, \beta_1)$ in this manner is a valid alternative. Of course ultimately we want to be able to talk about $C > 2$ colour types. There are at least two ways to achieve this goal from this point.

The first method is to simply generalize the above argument completely. Just as the above idea is inspired by work with infinite urns, the generalization we now suggest is also well-known in that context [36]. This time, instead of using $p_0 \sim \text{Beta}(\beta_0, \beta_1)$, we assume $(p_0, p_1, \ldots, p_{C-1})$ is Dirichlet with parameters $\beta_0, \beta_1, \ldots, \beta_{C-1}$, where each $\beta_i = \alpha_i + M\delta_i$. Then, by calculations analogous to those above for the $C = 2$ colour case, we can quickly show that this generalized conditional view of a finite urn leads to the same stationary distribution. The derivation is straightforward and completely analogous to that presented above, so we omit it here.

The second approach is to focus on a particular colour, say colour 0 without loss of generality, and consider the marginal distribution of having S_0 slots of colour 0 out of the M, versus any other colour. That is, essentially we think of any colour other than 0 as just some meta-colour type, say 1. We must remember of course that this colour 1 has an initial weight that is the sum of all the initial weights other than that for colour 0. Similarly, its associated weight at any time is the sum of the weights of each colour other than colour 0. The technique works for urns in the set $\mathcal{L} \cap \mathcal{D} \cap \mathcal{C}$ because in this class, knowing S_0 determines the weight of colour 0, by definition of

$\mathcal{D}$, and also the weight of all the other colours combined, since in $\mathcal{C}$ the total weight of every state must be constant. Hence, if we are only interested in quantities associated with specific colours at a time, we can use this second idea to effectively reduce the discussion back to a 2 colour urn. In fact, note that if our interest lies in a primed system, this same technique should work for the larger class $\mathcal{L} \cap \mathcal{D}$ as well.

The value of these alternate views of our finite urn is that they potentially allow us to directly compute certain components like $q_i(M)$, $q_i'(M)$, $\sigma_{ii}(M)$ or $\sigma_{ii}'(M)$ using appropriate conditional expectation formulas. For example, we can calculate $E[S_0(M)]$ and $(E[S_0(M)S_0(M)] - E[S_0(M)]E[S_0(M)])$ easily knowing that the probability of colour 0 draws in a stationary finite urn are conditionally independent Bernoulli(p_0), and $p_0 \sim \text{Beta}(\beta_0, \beta_1)$. That is, under these conditions the random variable $S_0(M)$ given p_0 is binomial with parameters M and p_0. Therefore, we simply compute

$$\begin{aligned} E[S_0(M)] &= E[E[S_0(M)|p_0]] \\ &= E[Mp_0] \\ &= ME[p_0] \\ &= M\frac{\beta_0}{(\beta_0+\beta_1)} \\ &= M\frac{(\alpha_0+M\delta_0)}{(\alpha_0+M\delta_0+\alpha_1+M\delta_1)} \end{aligned}$$

and

$$\begin{aligned} &(E[S_0(M)S_0(M)] - E[S_0(M)]E[S_0(M)]) \\ &\quad = E[E[S_0(M)S_0(M)|p_0] - E[S_0(M)|p_0]E[S_0(M)|p_0]] \end{aligned}$$

$$
\begin{aligned}
&= E[Mp_0(1-p_0)] \\
&= ME[p_0(1-p_0)] \\
&= M\left(\frac{\beta_0}{(\beta_0+\beta_1)} - \frac{\beta_0}{(\beta_0+\beta_1)}\frac{(\beta_0+1)}{(\beta_0+\beta_1+1)}\right) \\
&= M\frac{\beta_0}{(\beta_0+\beta_1)}\left(1 - \frac{(\beta_0+1)}{(\beta_0+\beta_1+1)}\right) \\
&= M\frac{\beta_0}{(\beta_0+\beta_1)}\frac{\beta_1}{(\beta_0+\beta_1+1)} \\
&= M\frac{(\alpha_0+M\delta_0)}{(\alpha_0+M\delta_0+\alpha_1+M\delta_1)}\frac{(\alpha_1+M\delta_1)}{(\alpha_0+M\delta_0+\alpha_1+M\delta_1+1)}.
\end{aligned}
$$

But then it is easy to calculate both

$$
\begin{aligned}
q_0 &= \lim_{M\to\infty} E[S_0(M)]/M \\
&= \lim_{M\to\infty}\left[\frac{(\alpha_0+M\delta_0)}{(\alpha_0+M\delta_0+\alpha_1+M\delta_1)}\right] \\
&= \lim_{M\to\infty}\left[\frac{\left(\frac{\alpha_0}{M}+\delta_0\right)}{\left(\frac{\alpha_0}{M}+\delta_0+\frac{\alpha_1}{M}+\delta_1\right)}\right] \\
&= \frac{\delta_0}{(\delta_0+\delta_1)}
\end{aligned}
$$

and

$$
\begin{aligned}
\sigma_{ii} &= \lim_{M\to\infty}(E[S_0(M)S_0(M)] - E[S_0(M)]E[S_0(M)])/M \\
&= \lim_{M\to\infty}\left[\frac{(\alpha_0+M\delta_0)}{(\alpha_0+M\delta_0+\alpha_1+M\delta_1)}\frac{(\alpha_1+M\delta_1)}{(\alpha_0+M\delta_0+\alpha_1+M\delta_1+1)}\right] \\
&= \lim_{M\to\infty}\left[\frac{\left(\frac{\alpha_0}{M}+\delta_0\right)}{\left(\frac{\alpha_0}{M}+\delta_0+\frac{\alpha_1}{M}+\delta_1\right)}\frac{\left(\frac{\alpha_1}{M}+\delta_1\right)}{\left(\frac{\alpha_0}{M}+\delta_0+\frac{\alpha_1}{M}+\delta_1+\frac{1}{M}\right)}\right] \\
&= \frac{\delta_0}{(\delta_0+\delta_1)}\frac{\delta_1}{(\delta_0+\delta_1)},
\end{aligned}
$$

which are familiar expressions by now.

While these results are not new, this alternate way of viewing finite urns is nonetheless interesting. We note that, at least in this special case, we are able to derive limiting quantities directly without relating unprimed and primed systems, and without having to assume their existence at any stage. Perhaps this method can be extended to the larger set $\mathcal{L}$ with more work.

4.6 Discussion

The results of this chapter have extended those of Chapter 3 to the limit in a natural direction. Typically, urns are studied as their time index, say n, goes to infinity. In our finite urn, we can consider this limit, but we also have a memory parameter M, and it is quite reasonable to question the behaviour as $M \to \infty$ as well. It is a little surprising to find that the order of taking these limits is significant. This fact is quite plainly confirmed using a simple $C = 2$ colour finite urn that is compared with similar results taken from infinite urn theory.

Throughout the chapter the limiting expressions we derive are compared with their counterparts in the infinite urn context. It turns out that there are both similarities and differences. We show that the corresponding limiting scaled mean vectors from a finite urn and an infinite urn are actually identical, for instance. On the other hand, the corresponding limiting scaled covariance matrices are not the same. That is, we find that the truncation that occurs in a sequence of stationary finite urns parametrized by the memory M, leads to a limiting process with different covariance properties than that of a limiting infinite urn. Finally, all evidence suggests that the

limiting process in the case of finite urns is still normally distributed, a result already accepted in the case of infinite urns. The main difference in the limiting behaviour of finite urns versus infinite urns then we conjecture, is in the set of limiting scaled covariance parameters. The main contribution of this portion of the overall work is the explicit derivation of an appropriate limiting scaled covariance matrix for the finite urn setting.

Chapter 5

Urns With External Influence

In this chapter we are motivated by the goal of modelling a dynamic control system. This endeavour will require taking a more global view than we have so far. On a purely abstract level, we can think of our finite urn as modelling a sequence of decisions where the current decision is influenced by past decisions according to some predefined function. Each successive decision becomes feedback for future ones. Since the only feedback is the outcome of the decision itself, it is quite reasonable to imagine that it is generated instantaneously at the location of the decision point. We will call this type of feedback *internal* from this point forward. A model based on this information alone is fine for studying any system which evolves as a single decision point in isolation. However, some applications may involve a network of decision points, where a decision made at one point actually influences the outcome of decisions at many other points. This is exactly the situation of a dynamic ant routing algorithm. Routing decisions made at one node ultimately affect the decisions made at all other nodes. For our purposes, feedback not originating at a given node will be termed *external*. In some

applications it may take time for external feedback to propagate throughout the network, but we will assume that any such delay is negligible. In fact this is not much of a restriction since in any real implementation a well designed network of this type may place priority on all feedback message passing. Any time this is true our assumption that external feedback is available immediately following a given decision is valid.

It is difficult to explicitly take into account all external feedback for an arbitrary decision network of any appreciable size. Instead, in the model we are about to present, we assume all external feedback to a given node is amalgamated and represented by a single process. We imagine that the behaviour of this external process is partially influenced by the decisions at the node in focus, but also, a portion appears to act quite independently of the local node. To capture this behaviour we allow the external process, which we will call Z, to update at times independent of the times when draws are made from the urn, yet we insist that updates do take into account somewhat the history of recent draws. Still thinking of the process of making decisions at a single decision point as the urn, it is as if we describe an urn embedded now within a larger decision network. In order to model the situation we have described properly, our new embedded functional weight finite urn as we will call it, must include information about the history of draws as before, together with information about the external process and its feedback. The components of the stochastic process representing this modified urn will both drive and be driven by each other. Since we have already observed that the individual parts may evolve on different time scales in general, we will have to introduce a global measure of time.

5.1 Embedded Finite Urn Model

At this point we present an urn that extends the model in Chapter 3 with the addition of an external process and its associated feedback. As already noted, we allow the external process to change state at times other than the times when draws are made from the urn internally. For this reason we can no longer simply index time by the number of draws, as we did with our original urn. Moreover, we are interested in how the sequence of decisions can track the evolution of the external process in real time, hence in this new formulation we will use t to denote the continuous time nature of our process. To keep the analysis tractable, we will insist that the times of both internal changes, or draws, and external process changes are each the times of independent Poisson processes. We take the respective rates of internal and external transitions to be μ and ν. So overall some transition will occur with rate $\mu+\nu$, and by the theory of competing exponentials the transition will correspond to a draw with probability $\frac{\mu}{\mu+\nu}$ and an external process change with probability $\frac{\nu}{\mu+\nu}$. In analyzing this continuous time urn we will take the approach of first considering the discrete time process at the transition points. Once we solve this discrete time process it is straightforward to construct the corresponding continuous time process using standard theory of the relationship between a discrete and continuous time Markov chain.

In building our new process we start with $X(n, M)$, this time viewing the time index n as counting the total transitions as described above. As before, this component records the history of draws to provide internal feedback. Next, we include a new quantity, $Y(n, M)$, which has the purpose of tracking external feedback corresponding to each draw. $Y(n, M)$ is a vector with M components, $Y_m(n, M)$, each one registering the external feedback counterpart to $X_m(n, M)$. We assume each $Y_m(n, M)$ records

an integer state from a set of F possible values. Finally, we must include the current state of the external process, $Z(n, M)$, since it may change at any time independent of the sequence of draws. We assume that Z may take one of E states that reflect the status of the external portion of the decision network. Our complete embedded urn then, is the process

$$\begin{aligned}(X(n,M), Y(n,M), Z(n,M)) &= (X_0(n,M), X_1(n,M), \ldots, X_{M-1}(n,M), \\ &\quad Y_0(n,M), Y_1(n,M), \ldots, Y_{M-1}(n,M), \\ &\quad Z(n,M))\,.\end{aligned}$$

A state in this process becomes (x, y, z), where $x = (x_0, x_1, \ldots, x_{M-1})$ is the current value of $X(n, M)$, $y = (y_0, y_1, \ldots, y_{M-1})$ is the current value of $Y(n, M)$ and z is the current state of $Z(n, M)$. Just as we may view x as either a vector or its integer representation in base C, if preferable, we will consider y as either a vector or an analogous integer representation in base F.

In terms of transitions, there are now two different types to consider. The first type, which we call an internal transition because it is triggered by actions at the decision point, corresponds to making a draw and obtaining feedback. With this type, the value of x and y change, dependent on the current value of z. Conceptually, an internal transition is a two step process. First a decision is made by drawing a colour from the urn. Then the corresponding external feedback is determined based on the colour chosen and the current status of the external process. For simplicity, we assume that the external feedback is a deterministic function of the colour selected and state of Z. Note that while the function may be deterministic, the feedback

is random, since the component Z is a stochastic process on its own. We will call this function $f(i,z)$, where i is the colour chosen and z is the current state of Z. With this in mind, the probability of going from state (x,y,z) to $(\tilde{x}_{(i)},\tilde{y}_{(i,z)},z)$ where $\tilde{y}_{(i,z)} = (f(i,z),y_0,\ldots,y_{M-2})$ is

$$p_{(x,y,z),(\tilde{x}_{(i)},\tilde{y}_{(i,z)},z)}(M) = \frac{w_i(x,y)}{w(x,y)} = \frac{\alpha_i + g_i(x,y)}{w(x,y)}. \tag{5.1}$$

Equation 5.1 is essentially a generalization of Equation 3.1, where this time we take $w_i(x,y)$ to be the weight of colour i, and $w(x,y) = \sum_{i=0}^{C-1} w_i(x,y)$ to be the total weight of all colours, in the urn in state (x,y,z). As before we assume $w_i(x,y) \geq 0$ for each colour i and state components x and y to ensure that the weight of each colour in the urn is positive. Again we also insist that the total weight is not zero, i.e., $w(x,y) > 0$, for each state so that $p_{(x,y,z),(\tilde{x}_{(i)},\tilde{y}_{(i,z)},z)}(M)$ is always well defined. The current weight of colour i accounts for the current value of both $X(n,M)$ and $Y(n,M)$ in order to take advantage of the increased feedback available. It does so through generalized functional weight components, now $g_i(x,y)$.

The second type of transition occurs when the external process moves from state z to state $\tilde{z}$. We assume that these changes in the external process are governed by an $E \times E$ transition probability matrix which we will call $\Psi(x,M) = (\psi_{z,\tilde{z}}(x))_{z,\tilde{z}}$. That is, we have

$$p_{(x,y,z),(x,y,\tilde{z})}(M) = \psi_{z,\tilde{z}}(x). \tag{5.2}$$

Note that the dependence on x reminds us that each component of $\Psi(x,M)$ is really a function of the history of draws. This is required so that the external process, whose

purpose is to portray the behaviour of the rest of the network, can take into account the effect of local decisions.

5.1.1 Special Cases

Given the many parameterizing functions required for a proper description of a functional weight finite urn under the influence of an external process, it is much more difficult to methodically classify them. Still, it is helpful to highlight some ways in which we can be more specific with respect to items like $g_i(x, y)$, $f(i, z)$ and $\psi_{z,\tilde{z}}(x)$.

We have seen that when the total weight, in this case $w(x, y)$, is constant, expressions are often simplified. This is certainly true of the first type of transition probabilities. It is not clear, however, that this simplification alone allows us to say more about the form of the stationary distribution in this case. It is an easily identifiable case, so we make mention of it here.

Another simple special case is any embedded urn that removes the dependence of the external process on the history of draws. In general, we have described the external component transitions as functions of the current state of X. Recall, this was done to reflect the belief that the external portion of the decision network is influenced in part by local decision outcomes. It may be true, however, that this dependence is less significant compared with other aspects. We may even opt to not include this effect in the model if this is the case. Mathematically, it suffices to say that the external process transitions, $\psi_{z,\tilde{z}}(x)$, are not really functions of x in this case. This has the effect of uncoupling the explicit dependence of the components of $(X(n, M), Y(n, M), Z(n, M))$ in one direction. Transitions in which X and Y change

are still heavily dependent on the state of Z, but transition probabilities corresponding to changes in Z become independent of X.

Other special cases can arise based on how we define the relation between Y_m and Z. For example, arguably the simplest way to define Y is to say that it is the value of Z at the time of the m^{th} most recent draw. Hence Y records an exact history of Z, much like X records a history of choices. This is equivalent to assuming $F = E$ and $f(i, z) = z$. This approach simplifies the description of Y, but simultaneously places the onus on the function $g_i(x, y)$ to interpret the status of the external process and the implications on the weight contribution. Another possibility is to let $E = 2^C$, and view the external state, in binary form, as giving explicit information about whether the rest of the network is currently favourable to each colour or not. That is, the ith binary digits, $0 \leq i \leq C - 1$, in the external state corresponds to whether the status of the rest of the decision network is currently favourable or not favourable respectively from the point of view of selecting colour i. With this interpretation for state z we further take $F = 2$, and let $f(i, z)$ simply take the value of the ith binary digit of z. We will see that this type of formulation is especially suited to building measures that count, on average, the proportion of time that a favourable colour is selected.

Yet another case we will make use of shortly in our analysis is one in which the external process can only be in one of C different states corresponding to whether the only currently favourable or correct decision is of type i, $0 \leq i \leq C - 1$. We assume that the external feedback also reinforces this interpretation. That is to say, if Z is in state $z = i$, then $f(i, z)$ is 1 if and only if $z = i$.

The last configuration we present in this section is best understood as a generalization

of a special case of a standard urn in the class $\mathcal{L}$. Recall that members in $\mathcal{L}$ have an associated weight matrix, Δ, that specifies coefficients for a linear function $g_i(x)$. The entries of Δ give the weight contributions of the various colours given the history of decisions. We note that this contribution can be thought of on a slot by slot basis. That is, each time we encounter a component $x_m = j$, for $0 \leq m \leq M-1$, we add Δ_{ij} to the weight of colour i for $0 \leq i \leq C-1$. Now with this interpretation in mind, we add another dimension to the data structure Δ and consider a similar situation. Suppose we expand Δ to include individual components of the form Δ_{ijk}, where we now take Δ_{ijk} to represent the weight contribution to colour i when colour j is drawn with feedback k. Then we can use

$$g_i(x, y) = \sum_{m=0}^{M-1} \Delta_{ijk} I_{\{x_m=j, y_m=k\}}$$

in this case.

In the next section we will begin by addressing the problem of finding the stationary distribution of our process $(X(n, M), Y(n, M), Z(n, M))$ as described above. We note that once we have this solution we can also easily obtain the analogous result for the continuous time process $(X(t, M), Y(t, M), Z(t, M))$ if we so desire. Moreover, if our interest ultimately lies with the compositions $W(n, M)$ or $W(t, M)$, perhaps the most fundamental quantities regarding urns, it is straightforward to produce these from $(X(n, M), Y(n, M), Z(n, M))$ and $(X(t, M), Y(t, M), Z(t, M))$ respectively.

5.2 Embedded Finite Urn Analysis

As with our previous urn process, $(X(n,M), Y(n,M), Z(n,M))$ as it has been laid out is a discrete time Markov chain. Our first goal is to solve for its steady state distribution using standard methods. With this in hand we immediately have a corresponding continuous time result. More importantly, we then can build useful measures of the modelled system's behaviour in both discrete and continuous time.

Unfortunately, unlike the results of Chapter 3, the explicit form of the stationary distribution is not evident in general or for any special cases of interest this time. Here, we are forced to fall back on numeric calculations of the steady state distribution. The exception to this, of course, is any situation in which this new urn process is equivalent to our previous urn. This occurs whenever we completely restrict the effect of the external process, and specifically its feedback, on the sequence of decisions. There are various ways to accomplish this task. Perhaps the easiest is to force $g_i(x,y)$ to be a function of x only. The external feedback becomes irrelevant in this case. Another possibility is to define the external process such that it only has one state. This forces each component of $Y(n,M)$ to be the same and makes it impossible to update the urn differently based on any variations in external feedback. At any rate, under conditions whereby this embedded functional weight finite urn reduces to a regular functional weight finite urn presented earlier, we can invoke all of the previously derived theory for the component X, including the specific form of the stationary distribution and any further limiting results. While nothing is gained by using an embedded urn where our earlier urn model would suffice, this observation does point out that the former is a pure generalization of the latter.

As with the regular functional weight finite urn, our starting point will be the single step transition matrix of our Markov chain. Keeping in mind the possible transition types of the form $p_{(x,y,z),(\tilde{x},\tilde{y},\tilde{z})}(M)$ given in Equation 5.1 and Equation 5.2 we construct

$$P(M) = \Big(p_{(x,y,z),(\tilde{x},\tilde{y},\tilde{z})}(M)\Big)_{(x,y,z),(\tilde{x},\tilde{y},\tilde{z})}. \tag{5.3}$$

We will use $P(M)$ to find the steady state distribution

$$\pi_{(X,Y,Z),(x,y,z)}(M) = \Pr\{(X(M), Y(M), Z(M)) = (x, y, z)\}$$

for this urn. Since we will be typically forced to obtain the stationary distribution numerically, the main contribution of this section lies in how we use $\pi_{(X,Y,Z),(x,y,z)}(M)$ to design measures. The measures we present can be divided into two types based on whether the behaviour we wish to study is steady state or transient. For the first type we will use the discrete time stationary distribution directly, while for the second type we will condition on discrete time events and obtain a continuous time result.

We bear in mind in the following that although the external component Z must be tracked in this embedded urn process, our primary interest lies in the evolution of the decision and feedback portions, together just $(X(n, M), Y(n, M))$. Remember, our view is still at a single node within the decision network. We can think of the external feedback as essentially a statement reflecting the success of decisions made locally. So ultimately $(X(n, M), Y(n, M))$ together provides enough information to gauge how this local node is performing under the influence of the external effects.

5.2.1 Steady State Measures

As always, we might be interested in the long run proportion of weight of each colour in the embedded urn. To this end we can calculate

$$E[W_i(M)/W(M)] = \sum_{(x,y,z)} \frac{w_i(x,y)}{w(x)} \pi_{(X,Y,Z),(x,y,z)}(M). \tag{5.4}$$

This alone reveals basic behaviour since it tells us the expected proportion of time that each type of decision is selected. This is fundamental information for any application. Also, and along this same tack, it is not overly difficult to find

$$\begin{aligned} & E[W_i(M)/W(M)W_j(M)/W(M)] - E[W_i(M)/W(M)]E[W_j(M)/W(M)] \\ = & \sum_{(x,y,z)} \frac{w_i(x,y)w_j(x,y)}{w(x)^2} \pi_{(X,Y,Z),(x,y,z)}(M) \\ & - \sum_{(x,y,z)} \frac{w_i(x,y)}{w(x)} \pi_{(X,Y,Z),(x,y,z)}(M) \sum_{(x,y,z)} \frac{w_j(x,y)}{w(x)} \pi_{(X,Y,Z),(x,y,z)}(M). \end{aligned} \tag{5.5}$$

This complements the previous measure in the sense that it tells us something of the variability of the decision processes. We note that in most applications it is not sufficient only that the system make certain decisions for specific proportions of time, rather, it is equally important that the system moves smoothly when it does change. Highly variable and verging on hysteric changes are usually not acceptable.

Probing a different dimension we can also zero in on the information given in the external feedback. Think once again of our key motivating example: an ant algorithm for dynamic routing control. In this modelling application of an embedded urn the goal is for the decisions made at local node to continually fit in with the rest of

the network. In terms of our process $(X(n,M), Y(n,M), Z(n,M))$ we can say this is occurring when the external feedback, $Y(n,M)$, indicates desirable outcomes on average. In a well constructed model, for each possible colour i it should be clear which external feedback states out of the F, if any, reflect desirable choices from the point of view of all other external nodes. More specifically, suppose we let $\mathcal{F}_i$ represent the subset of external feedback states that are favourable when colour i is selected. Then suppose we evaluate

$$\rho = \sum_{(x,y,z):y_0 \in \mathcal{F}_{x_0}} \pi_{(X,Y,Z),(x,y,z)}(M). \tag{5.6}$$

Note that ρ gives an indication of how often a correct choice, according to the selection of favourable feedback of course, occurs. Note that in the expression for ρ we have highlighted slot 0 since we only need to judge the feedback once as it first appears.

5.2.2 Transient Measures

When an external process is involved, the important aspect is often not whether the sequence of decisions are correct, but rather, how quickly they approach a favourable situation when a change occurs in the external process. In this case, we desire a picture of how the sequence of draws adapts to reflect external changes beyond the control of the local node. As always, consider ant routing as an example of requiring this type of control through feedback. If the network is relatively dynamic such that viewed from a specific node all other nodes together are constantly changing, then the goal of the ant routing algorithm running at a local node is to respond efficiently to the continuously changing network status. Ideally, if changes occur externally that require

new routes to be taken from the local node, a good ant routing algorithm should make necessary adjustments swiftly and smoothly. In this subsection we introduce a couple of probabilities that can be used to give an indication of the evolution of our process during periods of change.

As suggested in the preceding paragraph, we wish to investigate our process during a period of adjustment resulting from a major external change. To accomplish this consider any embedded urn defined such that a single external transition can result in a complete change of the preferred colour. That is, focusing on 2 different colour types, say i and j, imagine that prior to the external change the external feedback points specifically to colour i and not to colour j, as a preferred route, while immediately after the change the external feedback identifies a different colour j and not i. For concreteness we assume that the external feedback is always only 0 or 1 depending on whether the colour drawn is bad or good respectively in light of the external state. Suppose now further that a stationary urn of this kind has recently experienced a period of the constant external state favouring i and not j, say z_i^*, during which time the last M draws have selected the colour $x_m = i$ for $0 \leq m \leq M-1$, with corresponding ideal external feedback $y = 1$. That is, just before the change we have $x = x_i^* = (i, i, \ldots, i)$ and $y = y_i^* = (1, 1, \ldots, 1)$. Now, imagine that the external process changes to the state favouring colour j and not i, say z_j^*, and we observe the evolution of the urn with the Z component once again fixed. Initially following the change, which we take to be our new point of referencing time, the probability of drawing colour i will most likely still outweigh the probability of drawing colour j based on the stored feedback. This is to be expected since the urn remembers drawing colour i, with good feedback, for some time after the external change. Eventually,

however, as newer feedback arrives we expect the urn to deduce, hopefully with high probability, that colour j is the best choice with respect to the new external state. As time moves forward we expect the urn to be transitioning between states that exhibit a preference for drawing type j over all others.

Ultimately, we would like to produce plots of this behaviour against time. Furthermore, we seek real continuous time on the independent axis in order to produce smooth curves of the transition. The measures to be plotted are probabilities capable of determining that the urn is taking steps to evolve correctly given the constants of the setting. We propose two different probabilities here. Both will be conditioned on the discrete n step transition probabilities beginning just after the external state change. These probabilities are obtained by taking the powers of a subset of $P(M)$ for this process. We only consider a subset because we insist that the process is restricted from changing external state. So beginning with state (x_i^*, y_i^*, z_j^*) the instant after the external process change and only allowing internal transitions of the form given in Equation 5.1 we can collect the all possible reachable states in a set. We can even think of this set as the state space of some auxiliary process. Viewed in this way, there are exactly $\frac{C^M-1}{C-1}$ transient states, and C^M recurrent states in the auxiliary process. Either way, we only include these states in a new single step transition probability matrix which we call $P^*(M)$. We then use $P^*(M)$ to construct an n step transition matrix matrix

$$P_n^*(M) = (P^*(M))^n.$$

From $P_n^*(M)$ we can obtain the probability of transiting from a state (x_i^*, y_i^*, z_j^*) to states of the form $(\tilde{x}, \tilde{y}, z_j^*)$ in exactly n steps. We will use $p_{(x_i^*, y_i^*, z_j^*),(\tilde{x}, \tilde{y}, z_j^*)}(n)$ to

denote this probability, as it is taken from the row corresponding to (x_i^*, y_i^*, z_j^*) and the column corresponding to $(\tilde{x}, \tilde{y}, z_j^*)$ of $P_n^*(M)$.

Next, if we are going to condition on being in certain states after exactly n steps, and we ultimately want continuous time plots, we are going to further need to condition on there being n transitions in time t. Luckily, we know that the arrival stream of internal transitions is Poisson with parameter μ. Hence we know that the probability of n transitions in time t is given by

$$e^{-\mu t}\frac{(\mu t)^n}{n!}.$$

With this common background work done, we can present our two probability measures. We will see that they are quite similar in the way they try to gauge the actions of the restricted urn.

The first probability attempts to gauge how quickly the process reaches a set of states that we determine to be good given the external state $z = z_j^*$. Abstractly, we define this set to be $\mathcal{H}_j$ for now. In any application of the model the set of good states must be explicitly specified or determined through a stated criterion. With $\mathcal{H}_j$ chosen, we calculate

$$p_{i\to\mathcal{H}_j}(t) = \sum_{(\tilde{x},\tilde{y},z_j^*)\in\mathcal{H}_j}\sum_{n=0}^{\infty} p_{(x_i^*,y_i^*,z_j^*),(\tilde{x},\tilde{y},z_j^*)}(n)e^{-\mu t}\frac{(\mu t)^n}{n!}. \tag{5.7}$$

As explained earlier, we expect to see $p_{i\to\mathcal{H}_j}(t)$ increase as t increases. The rate of increase and ultimate equilibrium level will be of particular interest.

The second measure we present is the probability of drawing the newly favoured colour j at time t, assuming a draw is made at that moment. In order to accept

this definition we have to observe that although draws only occur at specific discrete points in continuous time, it is possible to imagine making a draw from the embedded urn at any point in time. That is to say, the urn only changes state at discrete times, but it is entirely possible to contemplate the functional weights of the urn at any time. With this in mind we calculate the probability of drawing colour j at any real time t conditioned on each restricted state of the urn at the time as

$$p_{i\to j}(t) = \sum_{(\tilde{x},\tilde{y},z_j^*)} \sum_{n=0}^{\infty} g_j(\tilde{x},\tilde{y}) p_{(x_i^*,y_i^*,z_j^*),(\tilde{x},\tilde{y},z_j^*)}(n) e^{-\mu t} \frac{(\mu t)^n}{n!}. \tag{5.8}$$

Although this probability is different strictly speaking, we anticipate the same sort of rising behaviour with time. We will scrutinize the plot of $p_{i\to j}(t)$ in a similar fashion as the first measure, paying close attention to the rate of change and the point at which it levels off.

5.3 Examples

The description of the stochastic process representing our embedded finite urn is still quite abstract at this point. Certain critical parameters, namely those necessary for describing transition probabilities, can only be determined once we establish the underlying application to be modelled. In this section we suggest examples of using embedded finite urns. Of course a common element in all of the applications we present here is the need for modelling an external process. We will see that each of the following examples have been touched on before, in a more restrictive setting. It should be clear, however, why an embedded functional weight finite urn is a more appropriate model for the types of applications below.

5.3.1 Dynamic Ant Routing Algorithm

In some sense this example is what we have been striving to model theoretically in this work. Our embedded functional weight finite urn is practically tailor made to address this application. We look here at the workings of an ant routing algorithm at a local node, embedded within a larger routing network. We assume that the portion of the network external to the local node is significant enough that the load to any route accessible from the local node is dominated by the external part. Therefore, the goal of the routing algorithm at the local node is to dynamically control the release of traffic in order to maximize coordination with the rest of the network. In this example we hope to see this behaviour through our model.

We will consider a small example for simplicity. We will take $C = 2$ and $M = 5$, since these values are large enough for illustration. Furthermore, as described we want to see the internal portion of our process effectively tracking the external portion in this example. If this is to occur at all, we should allow the internal transitions to happen more frequently that external transitions. Otherwise, the external process will vary so rapidly that internal update mechanism will never be allowed to learn the current external situations. This is to say, we will assume $\mu > \nu$ here. In fact, it is really the ratio of the two values that is important since we intend to leave the actual base unit of time unspecified. For concreteness, we will take $\mu = 10$ and $\nu = 1$.

It is sufficient to describe an urn that is essentially a combination of some various special cases mentioned in Section 5.1.1. For instance, to start we will assume that $E = 2^C = 2^2 = 4$ and the state of the component Z always reflects the status of each route from the local node componentwise. That is, viewed in base 2, digit i of state z tells us whether route i is currently considered bad or good by the network. We

will take 0 to indicate bad while 1 will indicate good. Moreover, to complement this we imagine that $F = 2$ such that we think of $Y = 0$ as an incorrect colour choice, and $Y = 1$ a correct colour choice, given the colour drawn and the state of Z. More specifically, we take $f(i, z)$ to be 1 if and only if the i^{th} component of z is a 1.

Regarding internal transitions, we will also assume that there is a Δ data structure associated with this urn. This structure, as previously introduced, can effectively parametrize each g_i as

$$g_i(x, y) = \sum_{m=0}^{M-1} \Delta_{ijk} I_{\{x_m=j, y_m=k\}}.$$

With $C = 2$ and $F = 2$ we need to specify $C^2F = 2^2 \cdot 2 = 8$ components of Δ. To keep things clean and simple we will assume that weight is only added to the same colour as that drawn. That is, we assume that $\Delta_{ijk} = 0$ if $i \neq j$. This implies that 4 of the 8 components are immediately 0. The remaining 4 will be split into 2 groups of 2 depending on whether $k = 0$, meaning the feedback was bad, or $k = 1$, meaning the feedback was good. Unlike static ant routing in Section 3.5 where the preference of routes has to specified in the parameters reflecting the routes, here the dynamic external feedback performs this role. To best illustrate this, and check that the model is working adequately perhaps, we assume that the routes are equally bad when the feedback is bad, and equally good when the feedback is good. Specifically, we take $\Delta_{000} = a_0$, $\Delta_{110} = a_1$, $\Delta_{001} = b_0$ and $\Delta_{111} = b_1$. Typically we will set $b_i \geq a_i$ to be consistent with our interpretation that $y = 1$ is better feedback then $y = 0$.

Now, while this form of g_i is interesting in itself, we want to allow an additional wrinkle here. Our view of the urn is already on a slot by slot basis. We note that since the slots represent different points in time from the past, we are in a position

to downplay older information. The rationale for this is the assumption that recent feedback may more accurately reflect the current state of the local node within the network at large. So in fact, for this example we actually propose using functional components of the weight in the form

$$g_i(x, y) = \sum_{m=0}^{M-1} e^{-dm} \Delta_{ijk} I_{\{x_m=j, y_m=k\}}, \tag{5.9}$$

where d is a parameter that we can vary to affect this downplaying notion.

Given our comments on the update functions in the preceding paragraph, the specification of the external transition matrix is critical. We mention first that here too, we define a special case that we have suggested before. For this example we assume the external process, Z, transitions independently of the history of draws X. Recall that in general Z may depend on X, but here we ignore this possibility. We have already established that Z has only 4 states. We imagine that both routes may be in either state 0 or 1, but, we assume that one route is preferred in this example. Moreover, we imagine that the states $z = 00$ and $z = 11$, in base 2, are transition states such that the proportion of time spent in either is small. The remaining proportion of time is unequally split between the other 2 states, with the greatest portion going to the state that corresponds to the route we prefer. Suppose the external process transition matrix is

$$\Psi(M) = \begin{bmatrix} \epsilon & 1-\xi-\epsilon & \xi-\epsilon & \epsilon \\ \xi/2 & 1-\xi-\epsilon & \epsilon & \xi/2 \\ (1-\xi)/2 & \epsilon & \xi-\epsilon & (1-\xi)/2 \\ \epsilon & 1-\xi-\epsilon & \xi-\epsilon & \epsilon \end{bmatrix}, \tag{5.10}$$

where we take the parameters ϵ and ξ to construct the transition probabilities. Without loss of generality we will assume route 0 is always at least as good as route 1. We include a small quantity $\epsilon > 0$ for transitions which we want to be highly improbable, for instance when 2 binary digits of z change simultaneously. Otherwise, we use the parameter ξ to control the level of preference of the state $z = 10$ over the state $z = 01$, both viewed in base 2 expansion. We insist that $1/2 \leq \xi < 1$ so that there is always pressure to move away from external state 01 and toward external state 10. In fact, for our purposes we imagine that ξ will be closer to 1. We must also ensure $0 < \epsilon < 1 - \xi$ so that $\Psi(M)$ is always well defined. Looking carefully at the external process transition matrix in Equation 5.10, in which both rows and columns are taken to correspond to external states in the order 00, 01, 10, 11, we see that there is always a tendency to move toward state $z = 10$. Of course we desire this imbalance. In this example we wish to confirm that our embedded urn, which is a model for a dynamic ant routing algorithm, can determine a preferred colour type or route.

This formulation provides a straightforward framework to do testing, while at the same time minimizing the size of the state space we must consider. Here the number of states is just $C^M \cdot 2^M \cdot 2^C = 2^{12} = 4096$. There is no trouble calculating the stationary distribution directly for this Markov chain.

Even in this relatively simple model we have several parameters to potentially adjust. A change in any of α_0, α_1, a_0, a_1, b_0, b_1, d, ϵ and ξ can have an effect on performance. We want to demonstrate the use of both types of performance measures for embedded functional weight finite urns. First, for the steady state measures, we will set ϵ and ξ so that one route is preferred over the other, and investigate the effect of α_0, α_1, a_0, a_0, b_0, b_1 and d. Second, looking at our transient behaviour measures, ϵ and ξ do

not enter the picture. We will assume colour 1 was previously preferred, and then consider the events following a switch of the external process. Again we will look at the effects of the various other update parameters α_0, α_1, a_0, a_1, b_0 and b_1 and the downplay parameter d.

The first thing we want to check is whether or not the model accurately detects various external process behaviour. We will fix each of α_0, α_1, a_0, a_1, b_0, b_1 and d, parameters all related to the weight functions, and observe the effect of different values of ϵ and ξ, which dictate external process behaviour. For this test take $\alpha_0 = \alpha_1 = 1$, $a_0 = a_1 = 1$, $b_0 = b_1 = 4$ and $d = 0$. This ensures that the routes are treated equally and there is no downplay effect. On top of this let us suppose that $\epsilon = 0.02$. At this point we have isolated the single parameter ξ, allowing us to investigate it in the range $1/2 \leq \xi < 0.98$.

Looking at Figure 5.1 we see that indeed the model is capable of determining the best route for a given external process setting. As we increase the likelihood of route 0 being preferred over route 1, the probability of selecting route 0 increases as well. In fact the plot seems to reveal an almost linear relationship over the range of ξ examined. Initially, we set $\xi = 0.5$ so that there is no difference between the routes from the point of view of the external process. In this case there effectively is no preferred route, so distinction between $b_0 = b_1$ and $a_0 = a_1$ has no effect. The ant algorithm simply muddles its way along, sometimes choosing route 0 with either bad or good feedback and other times choosing route 1 with either bad or good feedback. On average, there is nothing to sustain the probability of choosing route 0 at a value greater than 0.5. Later, however, when ξ is greater than 0.5 the algorithm chooses route 0 with good feedback more often than not. The larger the value of ξ the more

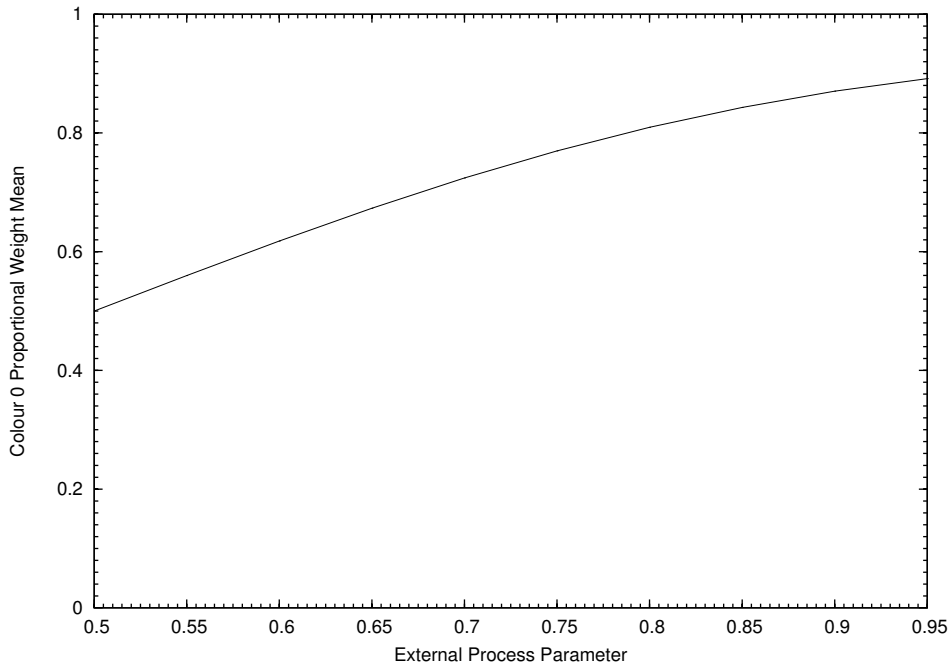

Figure 5.1: Probability of drawing colour 0 versus parameter ξ.

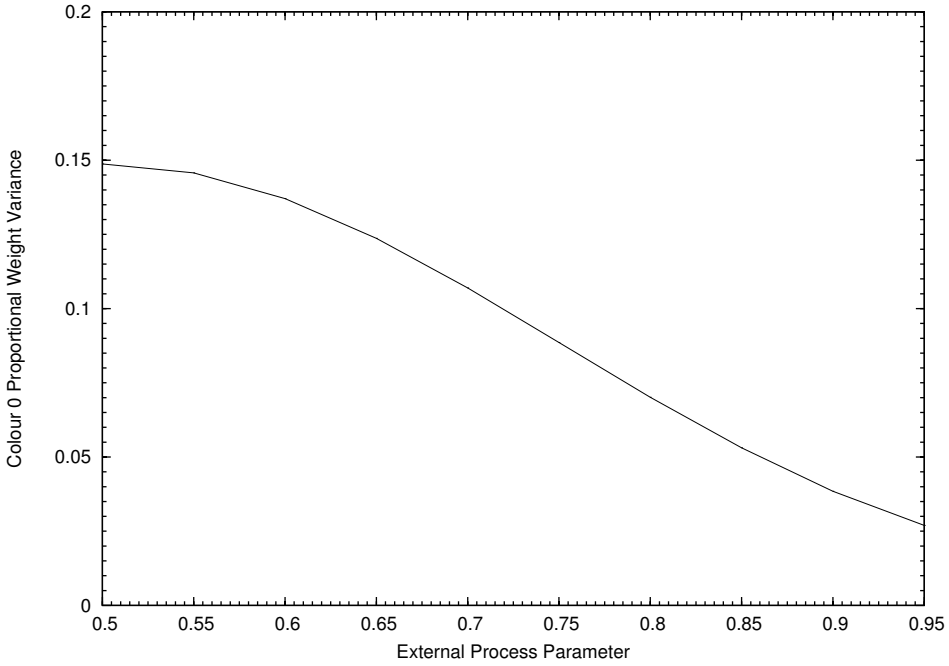

Figure 5.2: Variance of the probability of drawing colour 0 versus parameter ξ.

likely this occurs. This leads to the increasing trend we see depicted.

At the same time, Figure 5.2 confirms that the increasing preference of route 0 is increasingly more stable as well. Intuitively this makes sense. When the preference of one route over the other is less distinct, the algorithm will flip flop back and forth between the two. As the distinction is made more clear, the algorithm is able to settle on the preferred route more decisively.

Next, let us fix the external process with $\xi = 0.8$. We also take $\epsilon = 0.02$ as before. This time we study the effects of some of the other parameters outlined above. Again with $d = 0$ for the moment, let us consider the other functional weight parameters. To keep things simple, for the remainder of this example we will always take $a_0 = a_1$

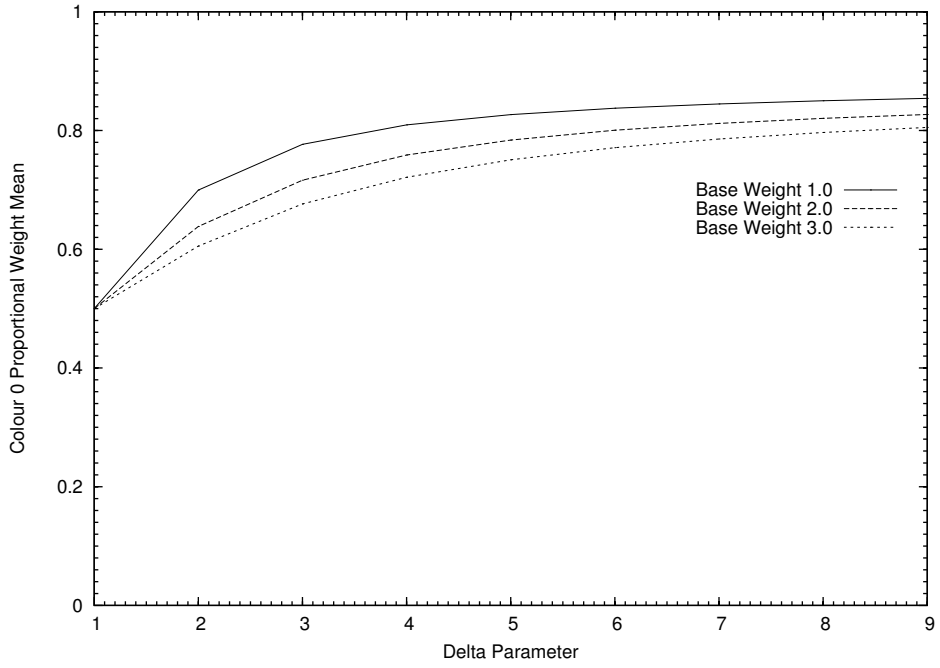

Figure 5.3: Probability of drawing colour 0 versus parameter b_0.

and $b_0 = b_1$. This can be interpreted as not artificially preferring one route over the other. The only differentiation between the favourability of the routes comes from the external process settings. Moreover, for now let us fix $a_0 = a_1 = 1$ and adjust only α_0, α_1, and $b_0 = b_1$. We consider each of the steady state measures this time.

We start by looking again at our most basic measure, the probability of choosing colour 0. This time we see that increasing the size of $b_0 = b_1$ relative to $a_0 = a_1$ has a clear effect on the probability of drawing colour 0. Initially of course, with $b_0 = b_1 = a_0 = a_1 = 1$ there can be no distinction, but as the size of $b_0 = b_1$ grows in Figure 5.3, with $a_0 = a_1 = 1$ fixed, so does the chance of selecting route 0. The rate of increase is steepest initially and then levels off to almost 0. That is, beyond a certain relative size increasing $b_0 = b_1$ further has little effect. This is due to the existence

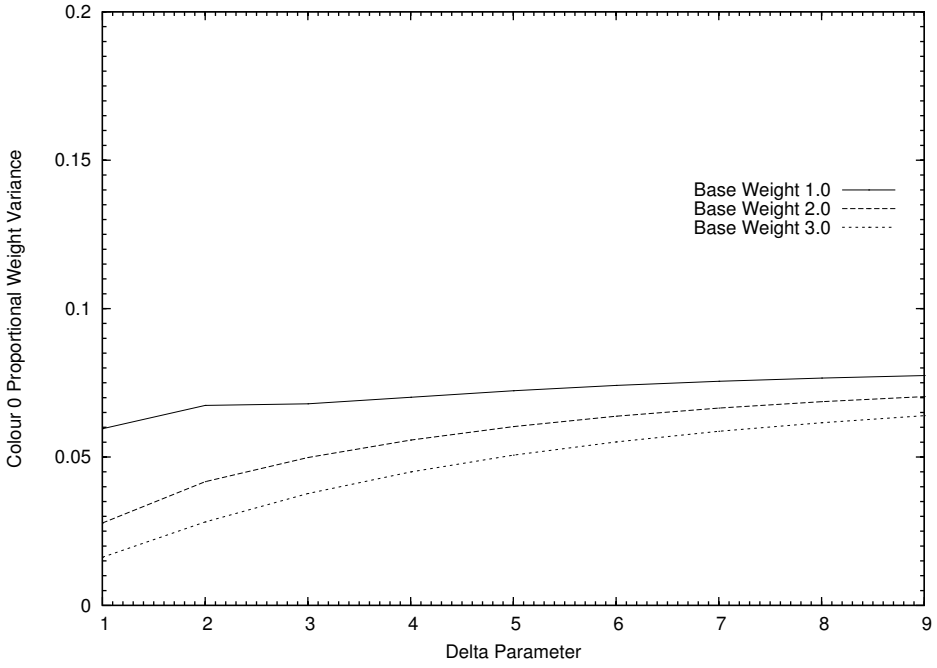

Figure 5.4: Variance of the probability of drawing colour 0 versus parameter b_0.

of base weights, which ensure there is always a chance of drawing even unpreferred colours. The ultimate point at which the curve levels out is dependent on α_1, which in this example is the same as α_0. To illustrate this control over the diminishing return of the size of $b_0 = b_1$ we have actually plotted 2 different values of $\alpha_0 = \alpha_1$ in Figure 5.3. We see that if we increase the size of the base weights, the probability of choosing route 0 is bounded further away from 1.

Another nice feature of the model is that increasing the relative size of $b_0 = b_1$ over $a_0 = a_1$ does not dramatically increase the variance of the probability of selecting our preferred colour. This fact can be seen in Figure 5.4. In fact, for the base weight values explored the variance is almost constant beyond $b_0 = b_1 > 3$. From a design perspective this is good news since it suggests we can adjust the relative size of the

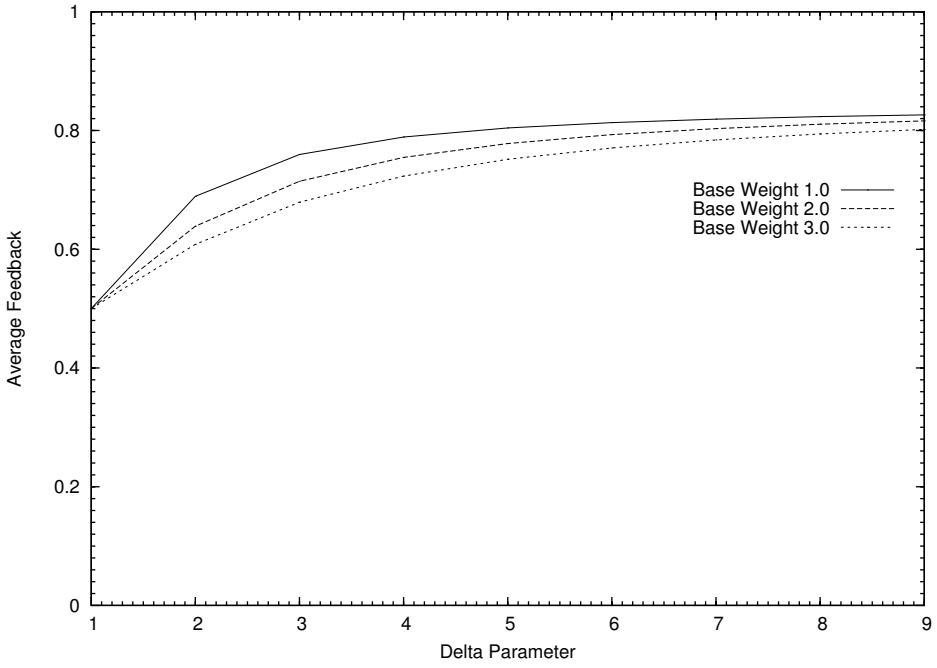

Figure 5.5: Average feedback in slot 0 versus parameter b_0.

parameter reflecting good feedback versus that reflecting bad feedback for a given route without worry of pushing the algorithm into an unstable situation.

Finally, here we also consider the expected average feedback measure introduced in Section 5.2. Recall that values of this measure approaching 1 are designed to imply better overall choices made by the modelled algorithm. Figure 5.5 shows that increasing $b_0 = b_1$ relative to $a_0 = a_1$ is also advantageous according to this measure. As before, initially there is no differentiation between the routes, so the external feedback is completely random and hence equally likely on average. For this reason the plots all start out at 0.5. As $b_0 = b_1$ increases though, the average external feedback indicates that better choices are being made for the external process settings. Note that as with the probability of drawing colour 0, the curves level off after a while. We

have already explained one reason for this before. Given the non zero base weights, there is always a chance of making incorrect decisions for the current external process state. Moreover, the larger the base weights, the further away from 1 we can expect to find the average feedback. This point is confirmed in the figure. Unlike the upper bound in Figure 5.3 however, here it seems that even with arbitrarily small base weight values these curves will not approach 1. This observation can be explained after remembering that the external process is free to change the preferred route from time to time. During periods of change, further incorrect decisions are possible. Hence, we never expect the average feedback to be arbitrarily close to 1 in this setting.

Next we want to consider the potential effect of the downplay parameter. To this end, we now fix $b_0 = b_1 = 4$ in addition to the constant values of all the other parameters, and release d for study.

Figure 5.6 depicts the probability of selecting route 0 for increasing values of the downplay parameter from $d = 0$, which we have encountered before, up to $d = 1$ in step sizes of 0.1. The plot is perhaps a bit uninteresting. Unlike other comparisons so far, the downplay parameter does not seem to have much of an effect on this basic measure of performance. In fact, all we want to see for the moment is that it does not have too much of an adverse effect. We will see later that the introduction of the downplay parameter has more to do with the rate of change of the algorithm than any steady state performance.

For completeness we do check the variability of drawing colour 0 with changing downplay parameter values. Figure 5.7 confirms that changing d has little effect on the variance of the expected weight of colour 0.

Finally, we also consider the average feedback under different downplay parameter

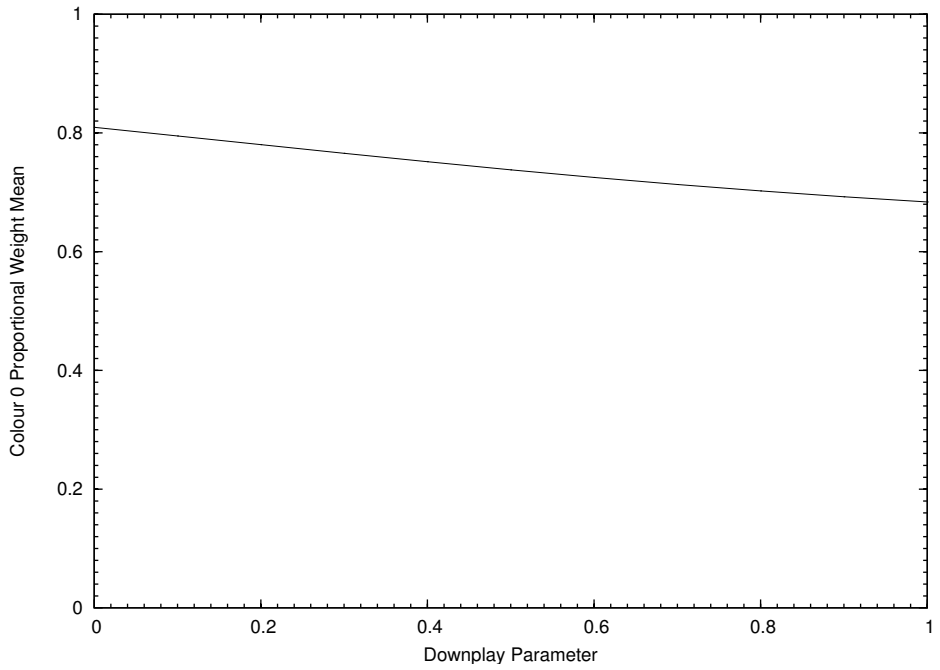

Figure 5.6: Probability of drawing colour 0 versus parameter d.

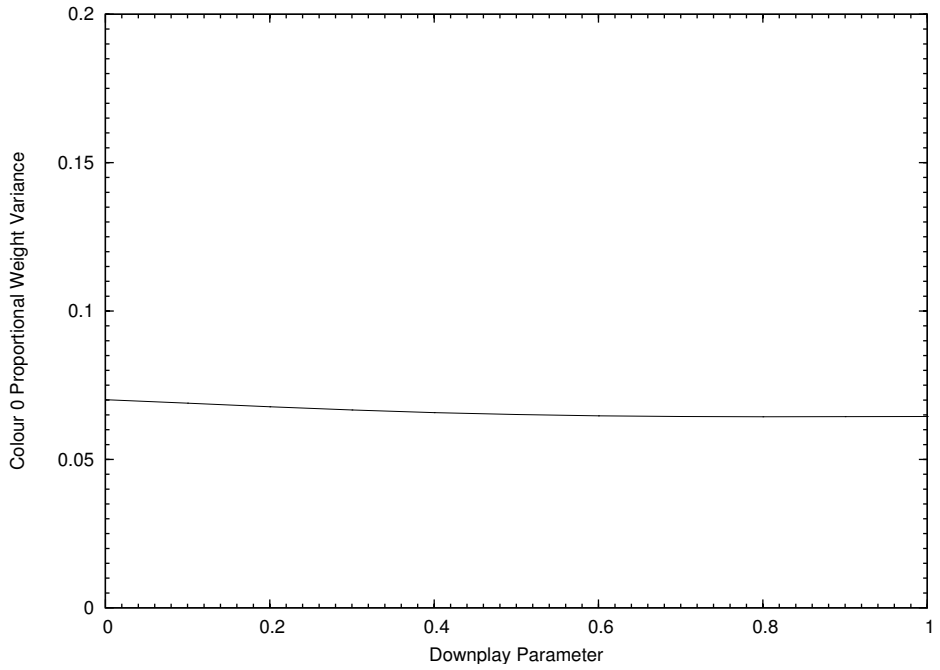

Figure 5.7: Variance of the probability of drawing colour 0 versus parameter d.

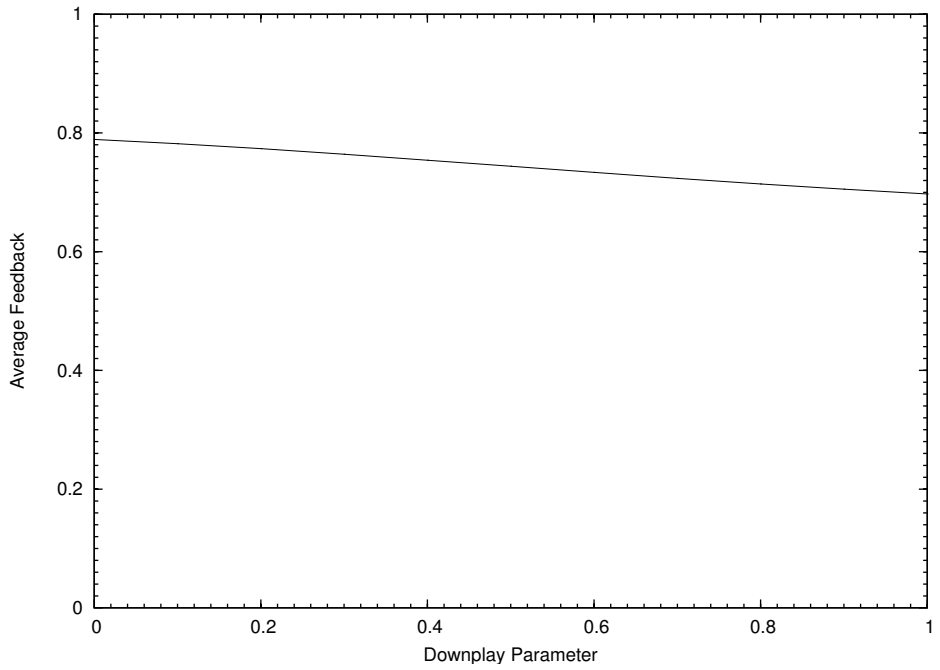

Figure 5.8: Average feedback in slot 0 versus parameter d.

settings in Figure 5.8. As with the probability of selecting route 0, there is very little change in this measure. There is a slight, almost constant, decrease but nothing dramatic enough to restrict us from using the downplay idea. In the next set of plots, in which we look at transient measures, we will see how downplaying the weight contributed by slots with older information can be an advantage.

For the remainder of this example we consider proposed transient measures from Section 5.2. It is important to look at transient behaviour in addition to steady state performance. The ant-based routing algorithm we are exploring here operates in a dynamic environment. We have explicitly modelled a random external process to ensure this fact. One of the greatest strengths of an ant algorithm in this setting is its ability to adapt and change quickly toward an ever moving optimal routing target. At the outset, we may know and accept that this desirable characteristic is often in conflict with the ability to settle on the absolute best routing policy in the long run. That is, we are often willing to trade optimal steady state behaviour for favourable adaptation qualities. It is under this assumption that we continue our evaluation here.

We will consider the transient measures in the order they were presented earlier. We recall that both measures involve looking at specific probabilities over time, immediately following a significant change in the external process. Here we will take the change to be from state $z = 01$, which favours colour 1, to $z = 10$, which favours colour 0. If we like, we can imagine the external process has just made the switch directly, an event which is rare, but still occurs with probability $\epsilon > 0$. After the change occurs we assume Z remains the same so that we can sample the evolution of the algorithm. Only transitions that result in changes to the X and Y components

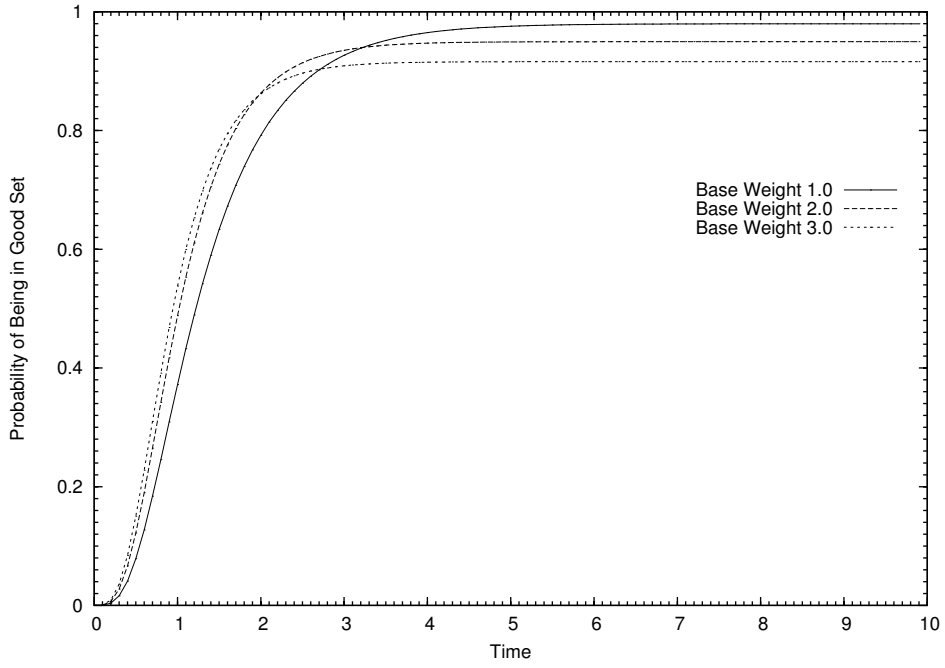

Figure 5.9: Probability of being in the good set for colour 0 versus time for various α_0.

are allowed. For this example the effective state space size for transient measure calculations is reduced to $\frac{C^M-1}{C-1}C^{(M-1)} = \frac{2^5-1}{2-1}2^{(5-1)} = 31 \cdot 16 = 496$. All possible transitions in this restricted view are collected into the matrix P^*, the powers of which are critical in determining our measures.

For the first measure, the probability of being in set of good states in terms of colour 0 or $p_{1\to\mathcal{H}_0}(t)$, we must specify the set of good states, $\mathcal{H}_0$. In order to do this systematically, we suppose that any state in which the probability of drawing colour 0 is greater than 0.6 is in $\mathcal{H}_0$. In general we could adjust the threshold of 0.6 to produce different sets of states, but for this example we will always take the good set to be defined in this way.

Figure 5.9 plots $p_{1\to\mathcal{H}_0}(t)$ using $a_0 = a_1 = 1$ and $b_0 = b_1 = 4$ for various values of $\alpha_0 = \alpha_1$. Note that the curves all start out at 0 since time 0 is the instance immediately after the external process change and typically the good set does not include the starting state, as is the case here. Moreover, we see that the probability of being in the good set eventually rises sharply and then levels off in each case, some distance away from 1. As always this is to be expected given the function of the base weights. The larger the base weights, the less chance there is of transitioning exclusively within the good set, however it is defined. Base weights ensure that exploration of lesser preferred routes continuously occurs. The portion of the plot where the curves rise dramatically is of great interest here. We will focus more on this phase in the following.

Before changing the values of Δ or the downplay parameter we look at our second transient measure, the probability of drawing colour 0 or $p_{1\to 0}(t)$.

Figure 5.10 plots $p_{1\to 0}(t)$ for the same values of $\alpha_0 = \alpha_1$ for comparison. Given the similarities between this second measure and the first, it is not surprising to find the curves have essentially the same features. The curves in Figure 5.10 also rise initially and then level off at values dependent upon the respective base weights. One noticeable difference here however, is that the curves all start off at values greater than 0. This is easily explained by the definition of $p_{1\to 0}(t)$. Even an instant after time 0 there is a non zero probability of drawing colour 0.

The previous two plots, Figure 5.9 and Figure 5.10, show the characteristic trend of our transient measures. Starting out small the probabilities go through a short period of rapid growth which then eventually subsides at some equilibrium level. We have seen how the base weights can alter this basic shape somewhat. Increasing the base

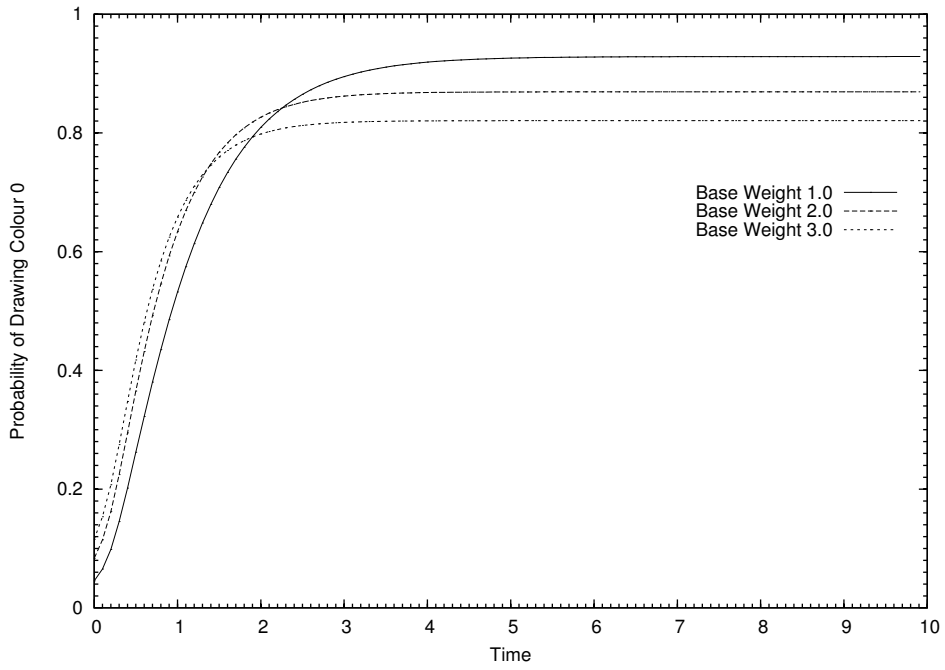

Figure 5.10: Probability of drawing colour 0 versus time for various α_0.

weights results in a slight increase in the rate of growth but ultimately leads to a lower equilibrium position. Ideally, we want to control this trade-off better. In the remaining experiments we show that it is possible to both increase the rate of growth and simultaneously maintain an acceptable equilibrium level using our defined model parameters. We demonstrate this in 2 different ways. The first method involves increasing the relative size of $b_0 = b_1$ compared with $a_0 = a_1$, while the second approach increases the downplay parameter. In order to confirm that ultimately the same equilibrium levels can be achieved, some tweaking of the base weights will be necessary. In reality the base weights should be adjusted anyway so that they always represent roughly an equal proportion of the total effective weight in the urn.

For the next couple of results we take $a_0 = a_1 = 1$ as always, and $d = 0$. We will vary $b_0 = b_1$ and note the effect on the rate of growth. We plot 3 different values of b_0 for both of the measures $p_{1\to\mathcal{H}_0}(t)$ and $p_{1\to 0}(t)$. We use $\mathcal{H}_0$ as previously defined.

Figure 5.11 plots $p_{1\to\mathcal{H}_0}(t)$ for $b_0 = b_1 = 2, 4, 6$. Respectively the base weights were adjusting to $\alpha_0 = \alpha_1 = 1.0, 5.0, 9.0$ through trial and error so that the equilibrium probability is approximately 0.85 for each curve. With the limiting levels pegged at the same amount it is easy to see the effect of the parameter b_0. Increasing the weight contributed when the external feedback is good versus when it is bad, can increase the rate at which the model evolves to our set of good states. In terms of the underlying application, this suggests that carefully increasing the ratio of $b_0 = b_1$ to $a_0 = a_1$ allows us to control the rate at which dramatic routing policy changes occur. Intuitively, the result is not surprising. It is well understood at this point that the difference in the build up of pheromone concentration is the force that evokes change in ant algorithms. Increasing $b_0 = b_1$ over $a_0 = a_1$ artificially amplifies the pheromone

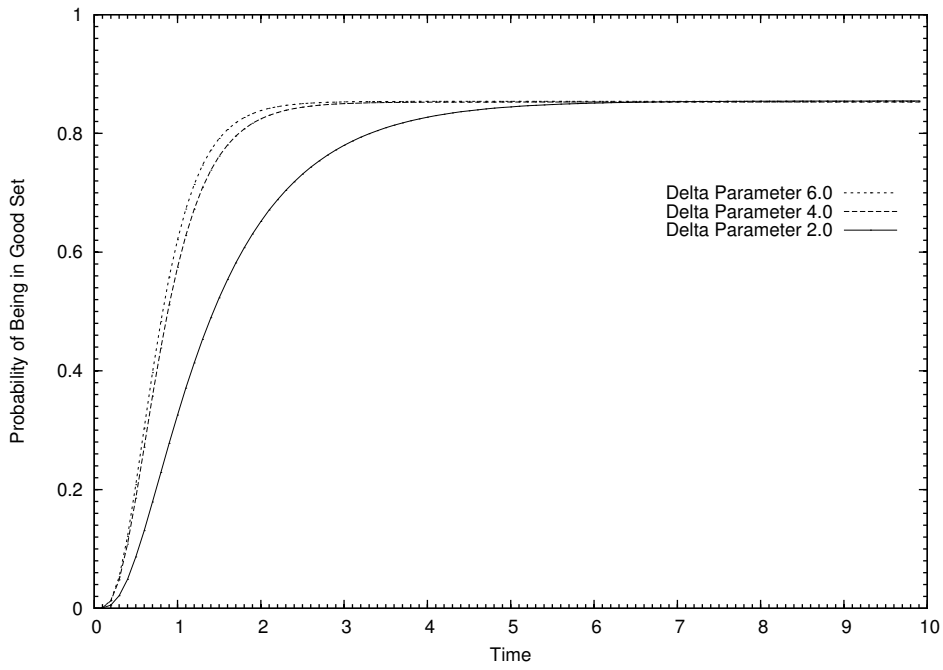

Figure 5.11: Probability of being in the good set for colour 0 versus time for various b_0.

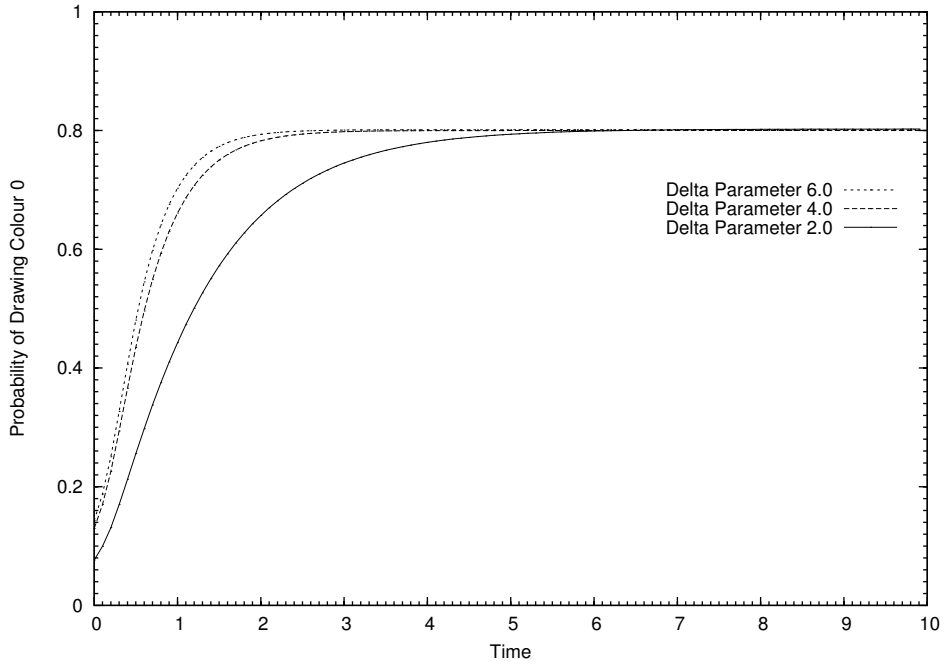

Figure 5.12: Probability of drawing colour 0 versus time for various b_0.

differential, leading to a faster uptake of the preferred route.

The second transient measure also confirms this notion. For the plots in Figure 5.12 we once again focus on $b_0 = b_1 = 2, 4, 6$, but this time take $\alpha_0 = \alpha_1 = 0.9, 3.5, 6.0$ respectively to ensure that the limiting probability of our measure $p_{1\to 0}(t)$ approaches approximately 0.80 in each case. We observe the same trend in the curves of $p_{1\to 0}(t)$. The reasoning remains unchanged from that above for the plots of our first measure, $p_{1\to \mathcal{H}_0}(t)$.

The last tests we run in this example take a closer look at the effect of the downplay parameter. Keeping our basic parameter settings the same we simply fix $b_0 = b_1 = 2$ this time and observe the result of changing d. Note that we take $b_0 = b_1$ on the

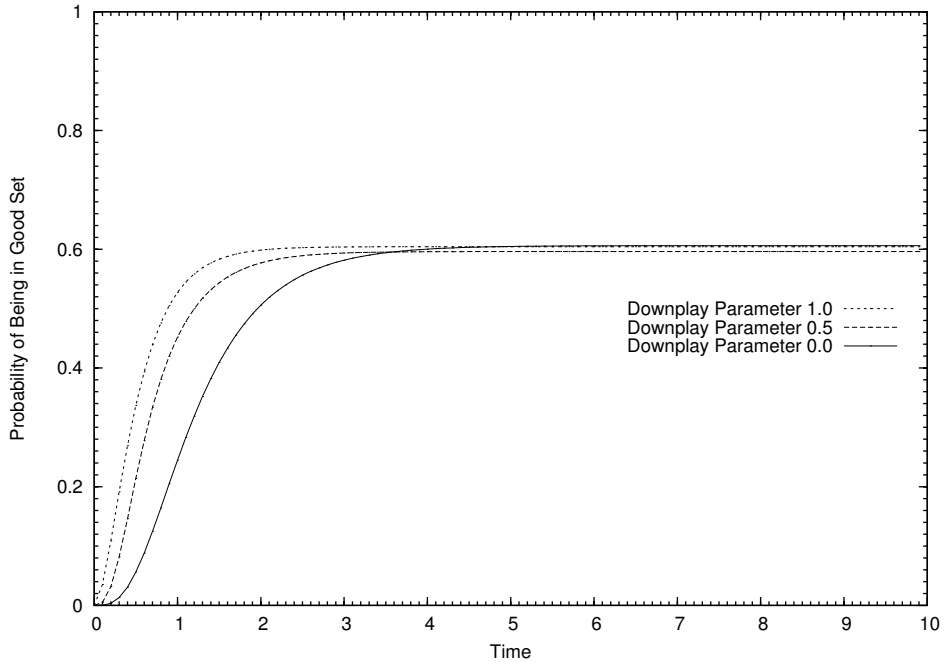

Figure 5.13: Probability of being in the good set for colour 0 versus time for various d.

lighter end of the spectrum so that the effect from this parameter alone does not unduly influence this experiment.

Once again we plot $p_{1\to\mathcal{H}_0}(t)$, this time varying d as $d = 0.0, 0.5, 1.0$. The results are graphed in Figure 5.13. We have again chosen to fix the equilibrium probability, this time at the value 0.60, by careful selection of the base weight parameters. For this test, $\alpha_0 = \alpha_1 = 1.8, 1.0, 0.7$ does the job for $d = 0.0, 0.5, 1.0$ respectively.

As in Figures 5.11 and 5.12, we notice that it is possible to take control of the rate of growth of the measure in a critical transient phase. Increasing d increases the aggressiveness of the change from a policy favouring route 1 to a policy favouring route 0. Put another way, this parameter is able to govern the aggressiveness of the ant

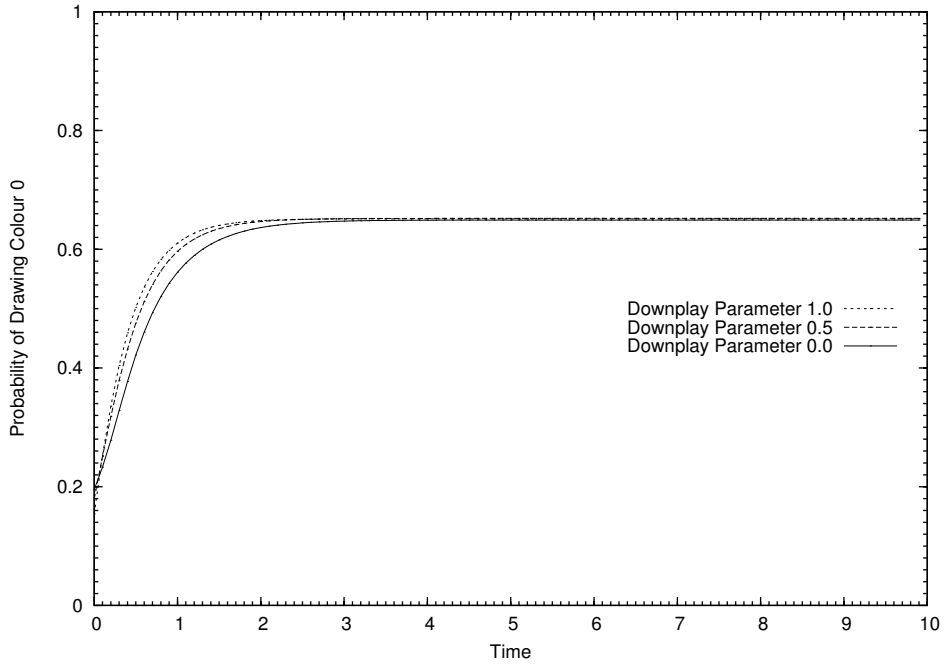

Figure 5.14: Probability of drawing colour 0 versus time for various d.

routing algorithm response to a required shift in routing policy. The effective reason for this is essentially the same as before. The difference in pheromone concentrations increases with increasing d. The mechanism that leads to this differential is not the same however. In this case, increasing d is like comparing fewer and fewer more recent slots at each iteration of the algorithm. It effectively takes fewer decisions, bad or good, for the algorithm to get a picture of what is going on. The differential is set up quicker, and the preferred route, in our experiment route 0, is able to outweigh the alternatives at a faster rate.

Finally, Figure 5.14 also complements our analysis of the role of the parameter d. Curves of $p_{1\to 0}(t)$ showing $d = 0.0, 0.5, 1.0$, with $\alpha_0 = \alpha_1 = 3.2, 1.3, 0.7$ respectively

to ensure approximately $p_{1\to 0}(t) \to 0.65$ in each case, illustrate a similar trend. Increasing d also increases the rate of growth of the probability of drawing colour 0 at time t in the crucial region in which the switch over is happening. The explanation for the observed trend is the same as that for the plots in Figure 5.13. This time however, we note that the curves are not as well differentiated as before. This is most likely a consequence of the starting points of the 2 measures. The probability of being in the good set is initially 0, whereas the probability of drawing colour 0 is initially bounded away from 0. At the same time, both sets of curves level off at approximately similar values. Hence, there is a larger range, over which the curves are differentiated to a greater degree, in the case of the probability of being in the good set, as compared to the case of the probability of drawing colour 0.

This example serves to validate our embedded functional weight finite urn. Interesting steady state and transient behaviour can be easily captured through a simple analysis of the included control parameters. We will make extensive use of the general concepts learnt here in Chapter 6 which deals with modifications to dynamic ant-based algorithms.

5.3.2 Other Applications With Random Feedback

Throughout most of this chapter we have focused on the ant routing application. The embedded urn defined above originally added the concept of external feedback to model some dynamic behaviour inherit in the system that is beyond the direct control of the local node. The existence of the external process creates an environment in which two identical decisions based on the same information stored in the state components X and Y can lead to different feedback. This is possible since the

component Z is free to change state at anytime, and essentially independently. The overall effect is that our embedded functional weight finite urn can, at our discretion, model highly random feedback.

Earlier in Section 3.5 we remarked that a user preference system where users leave personally tailored feedback, say through an exit survey, requires a more advanced urn process. It is understood that in order to handle the choice afforded each user in their feedback, a different more dynamic feedback process is needed. Our embedded urn, which takes into account random external feedback, is nicely suited for this purpose. With appropriate definitions of the external process and its feedback, this latter form of our urn can easily model a user preference system, and other similar applications.

5.4 Discussion

Leading up to this chapter, we only consider urns that evolve based on their own history of draws. While there are numerous applications displaying this type of internal feedback alone, here we have tackled a larger problem. Urns with only internal feedback take a local view by definition. This chapter broadens our focus to include an entire decision network. We have seen that there are interesting examples that require this view in order to be adequately modelled. We examined an ant routing algorithm for dynamic control, for instance. There we found it is critical to take into account external effects of the network since each node is continuously trying to realign itself with the others.

We formulated a new type of urn, which we called an embedded functional weight finite urn, that is a model for a single decision point within a larger decision network.

The process we described, which now monitors the state of the external decision points through another type of feedback, is in most ways a generalization of our original functional weight finite urn. Indeed, we even remarked that under certain conditions this embedded urn reduces effectively to an urn from Chapter 3.

With the increased complexity of the embedded urn model, we are not able to perform the same level of algebraic analysis that we have for the original functional weight finite urn. Instead, here we turned our attention toward developing measures that demonstrate the tracking ability of an embedded urn. That is, we constructed measures to analyze the behaviour of our complete process as its components evolve dependently on one another. Ultimately, this relatively straightforward model quickly highlights some of the pitfalls of such a dynamic system.

In the next chapter we move away from modelling and on to implementing ant algorithms for dynamic control. The basic concepts studied up to this point, and specifically in this chapter, will provide invaluable knowledge as we begin that endeavour.

Chapter 6

Modified Ant Algorithm Control Framework

So far we have focused mainly on producing mathematical models of ant algorithms. In this chapter we want to get back to the ant algorithms themselves. Here we provide our own implementation of an ant-based routing algorithm for dynamic control. As is done throughout this document, we will hold ant-based routing to the task of main motivating application for this chapter. It will be easy to see, however, that the concepts we present can apply equally well to any ant algorithm where the goal is control over a dynamic environment.

From the literature it is clear that ant-based algorithms bring intriguing levels of decentralization and adaptability, for instance, to the routing problem. However, for ant-based algorithms to be taken seriously for routing in data networks, they must be demonstrably reliable and perform at least as well as existing algorithms. Current proposals have limitations that keep them short of this mark. Take for example the

general limited control of responsiveness to dynamic traffic, or perhaps even a more basic problem, the potential for loops in data packet routes. These drawbacks should be resolved.

Responsiveness in ant-based algorithms is controlled through the functions which update the pheromone tables, coupled with mechanisms to discover new routes. More aggressive updates make larger changes to the pheromone levels and track changing traffic patterns more closely. However, this becomes undesirable as data packets start to track the transient congestion they themselves cause or they track short-lived traffic spikes, and the load balancing feature of ant routing becomes compromised. On the other hand, less aggressive updates can track real changes in the traffic load too slowly, as when part of a route that is not congested experiences a significant increase in load, resulting in many dropped packets before the pheromone levels at the nodes sending packets to the congested nodes have adjusted. The Schoonderwoerd ant routing algorithm [40] and the Di Caro AntNet [16] are both based on the notion of positive reinforcement. This means that the intent of updates is to increase particular pheromone levels in the pheromone tables relative to others. Pheromone levels may still drop however, through the process of renormalization of the entries in the table rows. In order to reduce a high pheromone level, ants must discover low probability routes and build up the pheromone on these routes, a process that may be slow and is at best awkward to control. This phenomenon is referred to as *routing lock* in [38].

Moreover, any good routing algorithm should guarantee loop-free routes for data packets, or at least guarantee that routes with loops are negligibly short-lived. To our knowledge no ant-based routing algorithm does this. The reason for this is that data packets follow next hop routing tables but the tables are not based on shortest

path routing. Thus, there is no built-in mechanism to avoid loops. Moreover, data packets and ants both use the same tables to choose next hops. In our view, this couples the behaviour of the data packets too closely to that of ants, and contributes to the awkwardness in obtaining the desired level of control of responsiveness in data packets.

In developing analytic models, specifically in Chapters 3 and 5, we have actually strayed somewhat from what might be considered the standard formulation of an ant algorithm. Whether or not any deviations were required to make the analysis tractable, or they were more clearly enhancements with a specific purpose, we will soon see that they suggest modifications that are advantageous in ant algorithm implementations.

It turns out that a fundamental modification, arising from the fact that our models are based on finite urns, has many useful implications for real world ant algorithms. We recall that an urn is a good base for modelling ant algorithms because of the way it incorporates iterative feedback into a form of collective memory. Even a carefully parametrized infinite memory urn is probably a good model for ant algorithms without an emphasis on evaporation. We choose however, to focus on finite urns, as they provide a mechanism that can be thought of as evaporating pheromone, which is a very important aspect of ant-based algorithms in general. In our model the finite urn contents still represent the evolving collective memory, but in a different way. As we point out in Chapter 3, we actually exploit the increased structure in our functional weight finite urn to take a different view of the way the history of feedback is stored and used. For starters, the slots of the finite urn are ordered. This is in contrast to standard ant algorithm memory. Typically, once information is incorporated into

the collective memory of an ant algorithm it is no longer possible to track its origins. With the slotted structure of our urn process we retain the ability to reevaluate each update as time moves forward. This is related to another feature that we also emphasize earlier. We view the urn contents, in terms of the weight of each colour, much more abstractly. It is a subtle but important difference. We do not imagine that the slots contain the actual balls of the urn, rather, the slots hold information about what draws have been made recently. From this information, at any time we can construct the weight composition of the urn through specified functions.

Translating these ideas into the implementation domain, we start to think of updates being stored in a database rather than directly incorporated into a pheromone table. This is an interesting development since one current limitation seems to lie in the fact that updates are memoryless in the sense that they are based only on current ant information. When the quality of a route changes to a given level, we do not necessarily want the update functions to respond the same way each time. Update functions are warranted which are based on more detailed information about the quality of a route, information that more accurately reflects how the route quality is changing over time. Furthermore, by building the pheromone table on demand from the database, we are in a better position to exert control over it. For example, we can drastically change the pheromone table from one iteration to the next without losing the individual feedback that has been collected. We can even build different pheromone tables for different purposes. That is, we may wish to have ants use one type of table and data use another. If a pheromone table is the only source of memory in the system, we are constrained in how aggressively we make changes to it. But with a database safely storing the history of feedback, any pheromone table built

upon it is theoretically freed of any constraints on its evolution.

In this chapter we suggest adding memory to the update decisions and argue that this is a crucial dimension for providing the means to obtain a desired level of control. We follow the philosophy of pushing as much of the routing complexity, in terms of the computational and memory requirements, onto the routers themselves. At the same time, we wish to keep the ants as simple as possible because this is a primary attraction of ant-based routing algorithms over more traditional routing algorithms such as distributed Bellman-Ford type algorithms or Dijkstra's algorithm. The finer granularity of the routing information dissemination in ant-based algorithms allows for finer control of the amount of overhead bandwidth used for routing.

To this end we propose that the routers should have more data structures than just a single routing table. In particular, routers should maintain databases of recent ant information. Moreover, we will discuss a compromise between the Schoonderwoerd and AntNet approaches to choosing where updates are applied, which is enabled by the databases. We believe that updates should be applied to nodes previously visited by ants, but that all ants should perform both forward and backward functions. This idea, too, will require an additional data structure to hold pending information. Finally, in order to harness the potential of the database concept we actually describe an implementation with separate routing tables for ants and data packets. This decoupling allows us to control the production of routing tables that better suit the differing goals of ants and data traffic.

6.1 Ant-Based Routing Framework

The modifications we propose are enabled by having at each router, in addition to the usual next-hop routing tables, three further data structures:

1. A *database* of recent routing information,
2. A *deliverables array* which contains information that needs to be picked up by ants and delivered to other nodes,
3. A *route routing table* for data packets.

For each source/destination pair of interest, the database will maintain the information delivered there by the flow of ants. The database will maintain this information up to a memory of M, where M is a tunable parameter that may represent a maximum number of ants into the past or a maximum time into the past. This database is used to directly support memory enabled updates to the routing tables for ants and data packets.

The deliverables array directly supports ants that perform both forward and backward functions simultaneously. The forward function of every ant will be to collect information about the route it travels (which it chooses according the next-hop routing tables) and simply drop the information into the deliverables arrays at the nodes it visits. At the same time the backward function of an ant will be to pick up information of use to the destination node of the ant from the deliverables arrays at the nodes it visits, and to drop this information at its destination node. For a given node this array contains information to be delivered to all known sources with this node as the destination. Updates are applied at source nodes when this information is delivered

to them. Typical information left by an ant is a measure of the delay experienced along segments of the path it has travelled so far, together with a time-stamp.

At a given node, the route routing table maintains information about the quality of routes from this node as a source node to every other node as a destination node. Each row corresponds to a given destination and each entry to a given route to that destination. We generically call the entries in this table pheromone, though the values may be very different and differently interpreted than the pheromone in the next-hop routing tables. We suggest that for a given destination node, information about the top C routes to that node be maintained, where C is a tunable parameter. Typically C need not be large and the set of the top C routes can vary over time. This table is obtained as a function of the database, as is the next-hop routing table for ants. Note that updates to this route routing table are in general different from updates to the next-hop routing table. Since this new table stores data corresponding to routes, not just next-hops, this data structure directly supports loop-free route selection for data packets. This is a welcome feature of this implementation.

It is important to note that the addition of data structures at the routers does not imply any increased overhead on the size of the ants. The ants can collect and drop off as much or as little information as is deemed necessary for satisfactory routing functionality. The quality of the information in the data structures will depend on the amount and frequency of information carried by ants.

6.1.1 Redefining The Collective Memory

Introducing a database results in a subtle but profound paradigm shift in our ant algorithm implementation. One of the fundamental concepts in ant algorithms is their ability to learn in a decentralized fashion by clever use of a shared memory. Typically this shared memory is understood to be represented by the pheromone table. Feedback is directly incorporated into the pheromone table and its effects are remembered because the update functions are constrained to evolve this table slowly. At the same time, the pheromone table of traditional ant algorithms fulfils another important purpose. It directly represents the distribution by which ants launch their exploratory function. We suggest that these roles are different and sometimes even in conflict with one another enough that their separation is advantageous. Really what is desired is memory of what has occurred, together with a cognitive mechanism that suggests intelligent actions based on this past knowledge. We propose moving the memory function of the pheromone table to a database, while retaining pheromone tables built from the database for the single purpose of providing an on demand summary or analysis of the choices in distribution form.

In this format, ants still choose where to explore based on a pheromone table, but they drop information they collect into an appropriate database. This in turn may trigger an update to a corresponding pheromone table, although we recognize that updates can be triggered at any time that is desirable. The functions that update pheromone tables are once again critical in determining how ants undertake exploration. The key advantage here is that these update functions can be much more flexible. They may change pheromone tables drastically from one iteration to the next, for example, a dangerous or at the very least delicate operation in the previous way of thinking.

Moreover, this blurs the distinction between adding pheromone or anti-pheromone, as it has been suggested in the literature, in a productive way. Essentially, at every iteration their is the possibility of producing an optimal pheromone routing table for the given feedback without delay. Previously, even if current feedback suggests a drastic change in routing policy is needed, we are at the mercy of slowly evolving pheromone tables.

Adding a database at each node complements any existing pheromone table by offloading the memory function. The database is merely a repository for the outcome of choices, while the update functions and resulting pheromone tables are the place where guided experimentation occurs. The database is more clearly identified as the memory, and any pheromone tables are simply functions of it. Without this decoupling we are forced to compromise on important tasks. We cannot experiment aggressively for fear of forgetting important feedback, yet we cannot avoid changing the memory else adaptation will never occur. In the new format, which adds a database, we can do both tasks better. We can remember as much as deemed necessary in the database, and still intelligently monitor and freely explore routes through the knowledge captured in the pheromone tables. Of course the success of the latter function hinges on the definition of routing table update functions. More will be said about these crucial functions later.

6.1.2 Increasing Ant Efficiency

When an ant travels from a source to a destination its experience provides information about the route *in the direction it is travelling*. Note that routers can, independently of the routing algorithm, monitor the congestion at all of their queues, and maintain

sliding window estimates of the average queue lengths for all of their incoming and outgoing links. Thus, we propose that ants should not experience any actual delay. Rather, they will simply collect the congestion information that the router is monitoring and be forwarded on an expedited control channel. The gathered information still needs to be sent back to the nodes it has previously visited, however. In AntNet, special backward ants are created to do exactly this task. It would be nice if these backward ants were also performing the functions of the forward ants, and vice versa, on the grounds of efficiency.

The deliverables array supports more efficient use of ants by enabling both forward and backward functions in every ant. As discussed, the deliverables array at a node holds information brought to it by ants from a given source (which may be an intermediate node along the ant's path from the ant's actual source) about a particular path to that node from the source. This information is not used to update the tables at the source node until another ant arrives at the node, destined for the source node, and carries that information back to the source node. Such an ant could have been generated at any node as long as the source node in question is its final destination. Note that such a backward travelling ant is not like the backward ants generated in AntNet in that they are not required to travel along the same route as the ant that dropped the information. In this way our *backward ants* can still carry out their forward function of route discovery.

This idea is based on the assumption that appropriate backward travelling ants are always readily being generated. If the goal is to transport routing information to the nodes that need that information as quickly as possible, then generating ants specifically for this purpose, as in AntNet, is optimal. However, when the frequency

of ant generation is sufficiently high, we may take advantage of the fact that there would have been a forward ant that could do the job in a sufficiently timely manner. Thus, we would like to create a trade-off between fast delivery of routing information for updates and efficient use of ants. The higher the frequency of ant generation the more efficiently we can use them.

To guard against information becoming too stale before it has a chance to be delivered, we propose that timers be maintained for information items dropped into the deliverables arrays. If the information in such an item has not been picked up by a ant before the timeout then the node will purposely generate an ant to carry the information back. An ant generated by this trigger will not carry out the forward duties of collecting and depositing information. It will retrace the route followed by the ant that originally collected and dropped the information, updating intermediate nodes with sub-path information as it travels.

Thus, the timeout controls the trade-off between efficient use of ants and latency of updates. On one extreme of the timeout value, namely timeout equal to zero, a backward ant is generated for every forward ant, and the latency of updates is essentially propagation delay, which we take to be negligible. The generation and functionality of ants is very much like that in AntNet [16] in this case. On the other extreme, when the timeout equals infinity, every ant performs both forward and backward functions. Compared to the zero timeout case, we need half as many ants to disseminate the same amount of information through the network, but the payload of each ant is larger on average, and the latency in the dissemination is increased. There are however, far fewer ants flowing in the network, relieving many routing decisions and other queueing delays associated with processing ants. One form of control

is to set the timeout value as high as possible while still maintaining acceptable average latency. What is an acceptable latency depends on how dynamically the traffic characteristics are changing.

Exactly how much information a hybrid ant (an ant that carries out both forward and backward functions) collects and drops off is up to the designer of a particular implementation of the algorithm. Every hybrid ant should at least collect information about the route it travels as it goes from its originating source to its final destination. Additionally, every hybrid ant may also collect and drop off information about any number of the sub-routes it travels, where a sub-route is one which either originates or ends, or both, at an intermediate node. For a route with l hops, there are in total $l(l+1)/2-1$ sub-routes. Each sub-route that an ant collects information about produces an information item, which will be placed in the deliverables array of the destination node of the sub-route. Note that the number of sub-routes about which an ant is allowed to collect information affects the average size of the payload of an ant and the speed at which the routing information is shared, but it does not change the way the timeout controls the trade-off between efficiency and latency.

6.1.3 Improving Control of Responsiveness

It seems neither necessary nor desirable for the decisions of data packets to be restricted to the information and mechanisms controlling the evolution of the ant flows. In our view, decoupling the pheromone information used by data packets from the pheromone information used by ants is a key to effective control of the behaviour of data packets. We recommend that the pheromone table for ants aggressively track

changing traffic conditions while we suggest that the pheromone table for data packets evolve more conservatively. We liken the discovery of routes to "beta testing" in software development. While the next-hop routing table can afford to be more experimental, the route routing table, used by actual data in the network, should be stabilized and tested like production code.

In this section we propose specific update rules designed to exploit the memory in the node databases and the decoupling of the control for ants from the control for data packets. These rules are presented here mainly as an illustration of the types of control we can exert on the evolution of the routing tables under this framework. Of course abstractly this framework supports arbitrary functions of the database.

Before stating the routing table update functions mathematically, we first outline some heuristic behaviour we want to capture in the list that follows.

- *Next-hop routing table for ants.* We have previously suggested that having pheromone in a routing table correspond to routes as opposed to outgoing links gives an immediate way of avoiding looping data traffic. This feature is highly desirable, and so it will be implemented in the route routing pheromone table used by data packets. While the separate routing table used by ants could be defined over routes as well, we specifically choose not to do this in the spirit of better exploration. We prefer to allow the ants to discover new and better routes when they become available. To this end, a next-hop routing table, much like a traditional pheromone routing table, will be used by ants at each node to select which outgoing link to follow.

- *Only an appropriate subset of the total database information used.* We assume

that the maximum number of entries stored in the database corresponding to any destination is M. From this total information we update both the next-hop routing table and the route routing table. While it seems reasonable to use all available information to update the table used by data packets, there may be an advantage in using only the most recent feedback in producing the table used by ants. We want to allow the ants to test a new lead quickly, so we include a parameter $M_0 \leq M$ that dictates how many database elements to consider when updating next-hop routing tables.

- *Routing tables inversely reflect delay estimates.* Entries in each row of a routing table, which all correspond to a given destination, will consist of a decreasing function of some net delay estimates. The larger the net delay, the smaller the corresponding entry. Certainly this alone is not new. However, here each entry will be updated individually according to some rule. This means, for instance, that each entry may potentially change at every update, and not only because of normalization as is typically done. In fact, with the addition of the database, there is no pressure now to normalize the rows of a routing table at all. If we need to view a row as a probability distribution for decision making purposes we can easily do that at the time. Finally, we do not wish any routing table entry to shrink below a set minimum threshold. We will ensure that an appropriate minimum pheromone level is always maintained.

- *Newer delay estimates preferred over older ones.* The database contains many feedback elements for each destination. Every piece of feedback contains a delay and a time-stamp. Comparing two delay estimates, one old and one new, it seems reasonable to treat the older one with more scepticism. We propose that

the contribution to the overall delay is downplayed as the age of the feedback increases.

- *Stale database information discounted.* The database stores information indexed by routes. It may happen that a particular route does not receive feedback for some time because it is undesirable. Once it becomes desirable again, we will want to include its new feedback, but there is little reason to include all of the old feedback pertaining to the period in which it was clearly bad. This outdated information would unnecessarily downgrade the preference for this route. To avoid this pitfall, we not only downplay delay estimates based on their age relative to one another, but we discount any delay estimate that is too old relative to the current time. This strives to ensure we always base our decisions on current, relevant feedback from the network.

- *Awareness of sudden and substantial changes.* One behaviour, characteristic of traditional ant routing algorithms, we want to overcome is their noted inability to react quickly when necessary. This is especially apparent when it becomes necessary to downgrade a route. Typically, when a route becomes bad the main way to decrease its associated pheromone concentration is via evaporation which occurs mathematically through normalization. The process can take too long, sending data on routes with known high delay long after the problem is detected. Under this framework we have seen that it is possible to drastically alter the pheromone levels associated with any route once the need for such changes are clear. If a currently preferred route returns significantly bad feedback all of the sudden, for example, we are now empowered to deal with the situation. We may continue to dispatch ants to the route to monitor the situation, while at

the same time we might adjust the data traffic routing table quickly to reduce its dependence on the now questionable path to the destination.

Next we want to translate the above goals into the mathematics of the update functions. Because the updates are functions of the information in the database, we begin by describing explicitly the type of feedback housed within.

The database at a given source node receives all feedback that makes its way from every possible destination node in the network. A bundle of feedback contains, at a minimum, information regarding the route taken from source to destination, an estimate of the delay traversing the chosen route and a time-stamp indicating the moment the information was deposited into the deliverables array at the destination. The route itself is specified by a list of all nodes visited from source to destination, including both the origin and the last node. We assume that the ants themselves are programmed to avoid loops so that routes are always valid. The feedback contains an overall estimate of the delay in general, which we do not require the ants to actually experience themselves. Instead, we imagine that at each visited node the ant receives an estimate of the delay that would be experienced by a data packet on the current link of the route. The time-stamp is taken to be the time of the feedback even though in reality the information comprising the delay estimate is taken at various points along the journey. For reasonable length routes this is a good approximation, especially in light of the fact that ants gather information without delay.

Ultimately, the feedback bundles arrive at the database and are stored according to their associated destination. We allow at most the last M collected pieces of information to be housed in the database for each destination node i. This limit is important as it ensures that information in the database is relevant when comparisons

are made between routes. We assume there are V nodes in the network, labelled 0 to $V-1$, so that there are $V-1$ possible destinations. It will be easy to think of the database as a $(V-1) \times M$ data structure where each entry has the form

$$(r_{im}, d_{im}, t_{im}) .$$

We also assume that a row of the database, corresponding to a destination node i, is ordered such that the entry in column m, for $0 \leq m \leq M-1$, contains the m^{th} most recent piece of feedback information in the database regarding trips to node i. Each entry gives the route taken to i, r_{im}, the estimated delay, d_{im}, and the time the information was created, t_{im}. For concreteness, we suppose the database is at node 0, and that every route in the network from node 0 to any other node is mapped to a unique route number. In this way, r_{im} takes values from this index set. For convenience, we will assume that the routes are ordered so that in the index set the first $C(1)$ correspond to destination 1, the next $C(2)$ correspond to destination 2, etc, and the last $C(V-1)$ correspond to destination $V-1$.

Furthermore, for a given destination i, out of the M pieces of feedback we may only be interested in those with a specific type of route taken to the destination. In particular, in a moment we will want to identify 2 different cases: routes with a common first link and individual routes. To facilitate the first type we will define $\mathcal{R}(i,k)$ to be the set of routes, out of the $C(i)$ that have destination i, which have first hop k. The second type, of course, is easily identified by a specific route. Both classifications highlight a set of feedback elements within a row of the database. In each of the respective sets we will require functions to identify specific indices from the set. In particular, in each row i we will want to know the column index of the next most

recent occurrence of the type from a given column m. For feedback identified by routes with a common first link, say k, let us call this next index $j_{0,ik}(m)$. In the case of database information pertaining to a particular route, say r, we will call the index $j_{1,ir}(m)$. For notational convenience, we will take $j_{0,ik}(-1)$ and $j_{1,ir}(-1)$ to be the first indices of the respective sets, when they exist. If there are no entries in any column greater than m in the database in row i with information regarding a given route with first hop k, we will take $j_{0,ik}(m) = -1$. Similarly, if there are no entries in any column greater than m in the database in row i with information regarding a given route r, we will take $j_{1,ir}(m) = -1$. Any reference to items of the database with the column coordinate out of the range $0 \leq m \leq M-1$ will return -1 for the route number, which of course means there is no route, ∞ for the delay estimate and 0 for the time-stamp. These conventions will ensure that the expressions to follow are well defined in all cases.

With the above notation laid out, we now consider the evolution of both the next-hop routing table, used by ants, and the route routing table, used by data packets. Mathematically, we imagine the next-hop routing table is an $(V-1)\times K$ matrix, where K is the number of outgoing links, which we will call P_0. The entries in each row of P_0, corresponding to each possible destination, contain the current amount of pheromone associated with each of the possible paths from the node. If it is not possible to reach a destination node using a particular outgoing link, the corresponding entry in P_0 will be taken to be 0. The route routing table, say P_1, performs a similar function, however its dimensions are different in general owing to the fact that the number of routes to a given destination is not necessarily K. In fact, for any moderate sized well connected network the number of routes from the local node under consideration to

any particular destination can be quite large. Additionally, as we have already noted, the number of routes to each destination i depends on i. For this reason, we assume that the maximum number of routes considered in any row of the route routing table is simply $C = \max_{1 \leq i \leq V-1} \{C(i)\}$. It is easy enough to take the pheromone levels in the last $C - C(i)$ positions of row i to be 0. In any actual implementation we may insist that the number of routes considered for each destination is fixed, but allow the set to change over time according to some rule. Of course, the non-zero elements in row i of P_1 are interpreted as pheromone levels associated with the respective routes to i.

Each time, say t, that an ant arrives carrying information about a journey to destination node i, the feedback is first inserted into the database. This requires dropping the oldest piece of information in row i and adding the new piece. Next, if the new feedback information specifies route r with k as its first hop, then the current values of $p_{0,ik}(t)$, from row i and column k of $P_0(t)$, and $p_{1,ir}(t)$, from row i and column r of $P_1(t)$ are updated according to

$$p_{0,ik}(t) = \sum_{m=0}^{M_0-1} u_{0,ik}(t,m) e^{-a_0 d_{im}} I_{\{r_{im} \in \mathcal{R}(i,k)\}} \tag{6.1}$$

and

$$p_{1,ir}(t) = \sum_{m=0}^{M_1-1} u_{1,ir}(t,m) v_{1,ir}(m) e^{-a_1 d_{im}} I_{\{r_{im}=r\}} \tag{6.2}$$

respectively, where the purpose of $M_0, M_1 \leq M$ and the scaling factors $u_{0,ik}(t,m)$, $u_{0,ir}(t,m)$, and $v_{1,ir}(m)$ will be explained in the following.

It is important to note that in each case these update functions are applied to each

entry in the row corresponding to destination i. That is, each of the K elements from a row of the next-hop routing table and the C elements from a row of the route routing table are given similar treatment at each iteration in terms of updating. This is in contrast to conventional ant algorithms which typically adjust only one entry and then re-normalize. By updating each element each time, according to the best information in the database, here our pheromone tables can evolve faster and more accurately. Some entries will be increased and simultaneously some entries will be decreased in the same update session. This very naturally incorporates a notion of anti-pheromone. We also explicitly do not normalize at each stage. Normalization can be done as necessary, when we need to view a row of P_0 or P_1 as a probability distribution when selecting paths for example, or at times when the computational load at the node is lighter.

In terms of the actual form of both Equation 6.1 and Equation 6.2, we see that a negative exponential function is used to translate a delay value into an amount of pheromone. Higher delay values lead to a smaller pheromone amount. While any decreasing function can be used, a parametrized negative exponential is simple, effective and has the added advantage that its range is bounded between 0 and 1. The parameters a_0 and a_1 serve to control the resulting size of the pheromone alloted over the range of delay values input. A larger parameter value reduces higher delay estimates more drastically. Since the pheromone values are ultimately compared against one another, larger values of either a_0 or a_1 lead to larger relative differences in the amounts in the next-hop routing table and route routing table respectively. Through analytic modelling we know that larger relative amounts of pheromone can result in

faster rates of reinforcement in periods of change. Hence, we expect parameter settings satisfying $a_0 > a_1$ to allow ants to aggressively test and adopt alternate routes when necessary, and data traffic to remain cautious, using only well tested paths.

Ignoring any scaling factors for a moment, we observe that these equations are analogous to Equation 5.9 from the main example of Chapter 5. The placement of the sum in the above expressions ensures that route types with more entries in the database are rewarded with a larger total amount of pheromone, as in Equation 5.9. This is an important ant algorithm property that works to continually reinforce choices that are consistently shown to be desirable. One slight modification above, however, is that the sums do not necessarily consider all M pieces of feedback information from the database. These control parameters enter into the framework so that we can adjust the memory of each routing table individually. Typically we will want to take $M_0 < M_1$ so that the next-hop routing table is quick to monitor and explore new trends in the database, while the route routing table remains more conservative owing to updates that take into account events over a longer time horizon. Most of the time, and assuming we truly have $M_0 \leq M_1$, it will make sense to simply set $M_1 = M$. Otherwise there is little sense storing all M pieces of information in the database.

As in the model of Chapter 5, we also choose to include a time based downplay factor in our framework. Multiplying the resulting pheromone amounts appearing in both the next-hop routing table update and the route routing table update, we have quantities that tend to reduce the amounts based on the age of the information. Specifically, these scaling factors are

$$u_{0,ik}(t,m) = e^{-b_0\left(t-t_{i,j_{0,ik}(-1)}\right)-c_0\left(t_{i,j_{0,ik}(-1)}-t_{im}\right)} \tag{6.3}$$

and

$$u_{1,ir}(t,m) = e^{-b_1\left(t-t_{i,j_{1,ir}(-1)}\right)-c_1\left(t_{i,j_{1,ir}(-1)}-t_{im}\right)} \tag{6.4}$$

respectively. The idea, already shown to be effective in the embedded urn model dynamic routing example, is that we should be able to control how much we value older information. Here we include this concept, in an expanded form that allows even finer control. In the above expressions we see that there are actually two types of parameters, which control the reduction based on age in 2 different ways. The first pair, b_0 and b_1, respectively alter the scaling that occurs depending on how old the set of information in the database for a particular type of route is relative to the current time. That is, if the entire set of feedback under consideration is old, it should all be viewed with scepticism. The second pair, c_0 and c_1, downplay successive delay estimates based on their age relative to the youngest among them. Again, as the information ages within the set it has the effect of decreasing the corresponding contribution of pheromone. Obvious simplifications are possible if either $b_0 = c_0$ or $b_1 = c_1$. Our earlier modelling experience demonstrates that increasing the damping according to age can result in better transient performance. This suggests that we will want to use $b_0 > b_1$ and $c_0 > c_1$ so that the next-hop routing table is liberal and the route routing table more conservative. The rationale remains as before. We want ants to aggressively pursue new leads, but at the same time, we want to handle data packet routing more cautiously. We can safely send ants to areas of changes in the network, but we do not wish to experiment haphazardly with actual data.

Lastly, we include an additional scaling factor in the route routing table update function. The purpose of this component illustrates the versatility of the framework built upon the database concept. Unlike the previous scaling factors, which penalize strictly based on time, this term potentially reduces the contribution of pheromone based on a simple trend observed in the delay estimates themselves. This time, we are on guard for increasing delay estimates in the set of feedback for a given route type, as they suggest a deteriorating situation that we want to avoid. On the contrary, we will not take special action if the delay estimates decrease, allowing only the normal reinforcement policies to take hold. We will make use of the notation $[x]_+$, which we take to mean x if $x > 0$ and 0 otherwise, to selectively monitor the increasing case and ignore the decreasing case. Specifically, we propose

$$v_{1,ir}(m) = e^{-f_1\left[d_{im} - d_{i,j_{1,ir}(m)}\right]_+}. \tag{6.5}$$

As usual, a negative exponential function acts to reduce the effective amount of pheromone when the current delay estimate, d_{im}, is larger than the next most recent delay estimate, $d_{i,j_{1,ir}(m)}$. The tuning parameter f_1 allows us to control the degree to which we take increasing delay estimates into account. Increasing f_1 leads to more aggressive reductions, while $f_1 = 0$ effectively eliminates this bias altogether since it would always be true that $v_{1,ir}(m) = 1$. Note that we have chosen to only include this last scaling factor in the route routing table update function. This fact actually highlights decoupling and the general belief that what is good for ants is not necessarily good for data packets and vice versa. In this instance, it seems prudent to have the route routing table, which governs the flow of data, back off quickly when feedback suggests a deteriorating route. On the other hand, there should not be a

problem sending ants, which are assumed to not increase congestion, to the troubled area to confirm what is taking place.

There is one final problem to address regarding the exploration ability of ants. A constant challenge facing most update schemes is striking the right balance between a tendency to continually reduce the pheromone level of a currently unfavourable route versus maintaining a sufficient probability of monitoring the same route for a change of status. In our framework we will simply insist that each value $p_{0,il}(t)$ remains above some minimum threshold at all times. This will ensure that there is always a chance ants can discover newly favourable routes. Interestingly though, we do not need to insist that route routing table entries remain above any level since data packets are not involved in the exploration process. We are free to let the amount of pheromone associated with a bad route approach 0 in the route routing table. This means, for example, we do not need to send a portion of the data traffic along routes that are known to be congested as a side effect of maintaining adequate exploration capability. This simple yet highly desirable feature is a consequence of the decoupling that is enabled by the addition of the database.

6.1.4 Distributed Ant Routing Algorithm

Given the previous lengthy discussion of update functions, it is clear that ultimately the behaviour of this kind of dynamic ant algorithm hinges on their exact specification. Care must be taken in selecting appropriate update procedures. Assuming the update functions have been defined however, the modified algorithm in the context of our framework is actually quite straightforward.

For completeness, we now formally outline our distributed ant routing algorithm in pseudo-code. Provided each node in the network runs such code, the desired global behaviour emerges out of the copies running in parallel.

```
Main() {
    t_cur := current time index;
    t_end := algorithm termination time;
    t_gen := time between ant generation;
    while (t_cur < t_end) {
        if (t_cur mod t_gen = 0) {
            Launch_hybrid_ant();
        }
        Check_for_expired_timers();
        t_cur := t_cur + 1;
    }
}

Launch_hybrid_ant() {
    Select_random_destination();
    while (NOT_AT_DESTINATION) {
        Select_next_hop();
        Collect_information_from_node();
        Load_deliverables();
        Move_to_next_hop();
```

```
    }
    Drop_collected_information();
    Unload_deliverables_into_database();
    Update_next_hop_routing_table();
    Update_route_routing_table();
}

Check_for_expired_timers() {
    for (ITEM in DELIVERABLES_LIST) {
        if (ITEM_TIMED_OUT) {
            Dispatch_backward_ant();
        }
    }
}
```

Note that `Update_next_hop_routing_table()` and `Update_route_routing_table()` are not specified here. We have already given detailed suggestions for their implementations earlier.

6.2 Simulation Analysis

In order to analyze our ant routing implementation we have written an event driven simulator capable of mimicking ants and data flowing on arbitrary network topologies. The simulator takes as input information about the network topology, data traffic

loading and parameters specifying the amount and frequency of ant activity. It also requires values for the size of sliding windows for delay estimates, the number of updates stored in databases and the various tuning parameters in the functions that update the routing tables.

At the beginning of each simulation run, there is a period of route discovery and pheromone table burn in, all before data traffic begins to flow. Initially, ants build routes by randomly choosing from the possible outgoing links at each step. This allows the simulator to discover routes, recording them in the appropriate route routing tables for later use. Eventually, once a sufficient number of routes have been found, ants make decisions according to the entries in the next-hop pheromone tables. It is important to note at this point that the best method for governing the dispatch of ants is still open for investigation. For simplicity here we send out our information gathering agents in waves, which occur at regular intervals. In each wave, each node is allowed to dispatch a single ant to a random destination node. This method is simple and quite feasible given the centralized nature of a software simulator, however, real world implementations will likely favour alternatives. Once data begins to flow, the pheromone tables are allowed to further burn in, and then ultimately, we start collecting performance data.

Since an implicit goal of this chapter is to study an ant algorithm's ability to control a dynamic link cost environment, we ensure there is opportunity and reason for adaptation. To this end, the simulator is built around a dramatic load switch over point. To this end, data packets are generated at each source node, always with a specified rate per destination node, under two significantly different loading schemes. We let the algorithm settle on the first loading scheme and then abruptly change to

the second loading scheme. Our analysis then focuses on the framework's ability to cope in the transient period around the time of the switch.

The simulation is coded such that various performance measures are taken at fixed intervals. Chief among these are specific real-time normalized pheromone levels from each of the routing tables. For comparison purposes, a set of routes are determined by specifying a source node, destination node and first hop, and then the corresponding cumulative pheromone levels are taken from both the next-hop routing table and the route routing table. Note that it can be the case that specifying a source node, a destination node and a mandatory first hop completely specifies a unique route. This is true within rings, for example, a very common network substructure. These values are viewed as probabilities and plotted in each case. It is then interesting to compare them to the transient plots in Chapter 5, from the analysis our dynamic embedded urn model, but also to each other, to evaluate the level of decoupling. At the same time, we also estimate the efficient use of ants. We do this by considering the average number of ant waves before collected information reaches the appropriate database together with the average number of backward ants dispatched. Recall that backward ants are only dispatched when information waiting in a deliverables array is about to expire. Finally, we also measure the network-wide average end-to-end data packet delay. This is done to ensure that our implementation maintains reasonable load balancing at all times, a type of control ant algorithms are known to demonstrate well.

Further information on specific details of the simulator can be found in Appendix B.

6.3 Examples

In this section we explore two different networks with our simulator. The first, a 4 node ring, is selected because it is simple to visualize, yet the results suggest how our ant algorithm framework effectively manages a single choice point. It is also the same type of topology underlying the examples already considered in our analytic models since a ring admits exactly two paths between any pair of nodes. The second example considers a real-world network, CA*net 4. This is currently Canada's most advanced research network. Most importantly for our purposes, it is a reasonable sized network with paths that are long enough to require multiple routing decisions to be made. This provides opportunity to investigate truly embedded choice points that are effected by traffic from various points in the network simultaneously.

6.3.1 Simulating Transient Behaviour at a Choice Point

To begin exploring our proposed framework we consider the network topology given in Figure 6.1. This simple, symmetric, example was specifically chosen so that we may anticipate the desired behaviour of our routing algorithm. We keep in mind, however, that despite its simple appearance, rings such as this are often the building blocks of many real-life network topologies. In fact, a 4 node ring is essentially a good proxy for any size ring since we can model 2 distinct paths between a pair of opposing nodes. By working to understand how our routing algorithm decides which way to send traffic travelling from a node on one side of such a ring to a node on the other we glean insight into how any choice point is handled.

We have already mentioned that we consider two different loading schemes and for

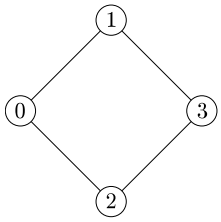

Figure 6.1: 4 node ring topology.

some measures, only a specific set of routes. In this case we will focus our attention on the route which is uniquely determined by specifying source 0, destination 3 and first hop 1. At times we may refer to this route as the path on the upper branch of the network and distinguish it from route 0-2-3, or the lower branch, as depicted in Figure 6.1. For simplicity, we imagine that all nodes transmit data to each of their neighbours at Poisson rate 10 packets/second in both loading schemes, with a couple of important exceptions. In the first loading scheme, node 1 transmits to node 3 at rate 120 packets/second while node 2 transmits to node 3 at the regular rate of 10 packets/second. After the switch, that is, in the second loading scheme, node 1 resumes transmission to node 3 at rate 10 packets/second while node 2 increases its transmission rate to node 3 to 120 packets/second. With the traffic specified in this manner we can anticipate, and hence verify, the working of the ant routing algorithm. Up to the switch point, the upper branch should be relatively congested, so we expect traffic from node 0 to node 3 to take the lower branch route. After the switch over, we expect the opposite to be true. That is, in the second loading scheme we want to see traffic from node 0 to node 3 using our highlighted route, 0-1-3.

Moreover, throughout this example we will assume that the data packet transmission time, which will include all processing time at the router, is fixed at 0.005 seconds. We

will also take the ant wave inter event time to be 0.4 seconds. We note that this ensures that on average the network contains only 12 ants/second, even assuming backward ants are generated 20% of the time, compared with 230 data packets/second. This implies a ratio of ant objects to data packets objects of roughly 5%. In addition, we assume that the payload of an ant is only a small fraction, perhaps 10%, of the payload of a data packet. Hence, the actual overhead incurred as a result of sending ants is more like 0.5% in this example. Finally, we will store up to $M = 40$ pieces of information in every row of every database, and we will collect performance measures every second.

In Figures 6.2 and 6.3 we present transient normalized pheromone values corresponding to the upper route for the next-hop routing table and the route routing table respectively. The values are taken from a single simulation run. In terms of the update function parameters, we take $M_0 = 5$ and $M_1 = 40$ and temporarily ignore downplaying with respect to time or discrepancy in consecutive delay estimates. That is, these plots were created using $a_0 = 400$, $a_1 = 800$, and $b_0 = b_1 = c_0 = c_1 = f_1 = 0$. The loading scheme switch occurs at 200 seconds, as measured from the point we begin taking performance statistics.

We immediately note that both tables have correctly identified the best route to use for traffic from node 0 to node 3 at all times. Before the switch over both tables indicate a strong preference for route 0-2-3, since it is the only alternative to route 0-1-3, while after the switch they both eventually settle on 0-1-3 as the preferred path. We notice that the general shape of the curves is reminiscent of the shape produced by our model in the routing example of Chapter 5. This is not a huge surprise given that the form of the ant algorithm implementation update functions here is explicitly

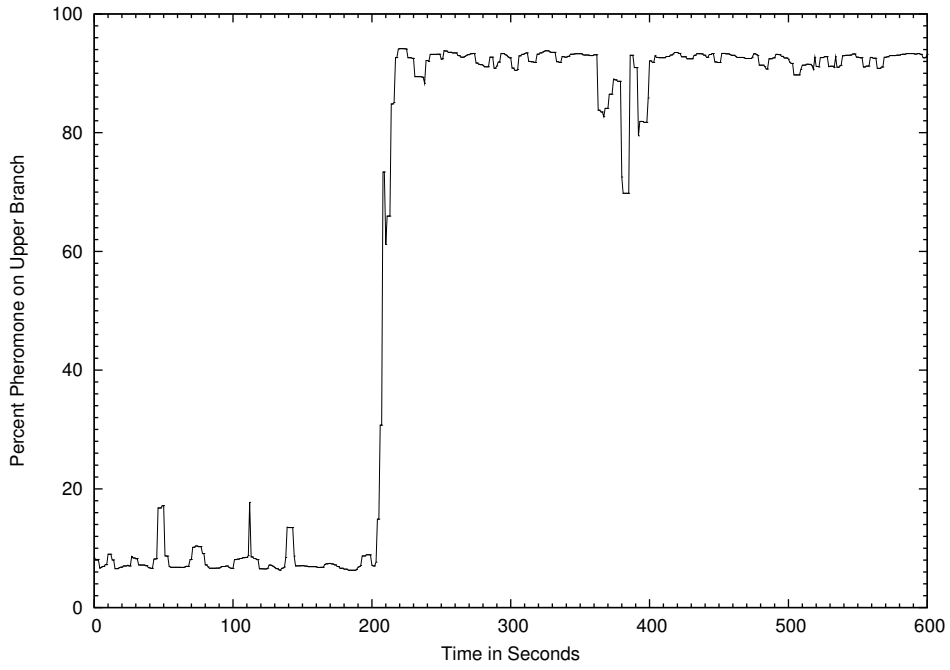

Figure 6.2: Proportion of pheromone on the upper branch according to the next-hop routing table with respect to time.

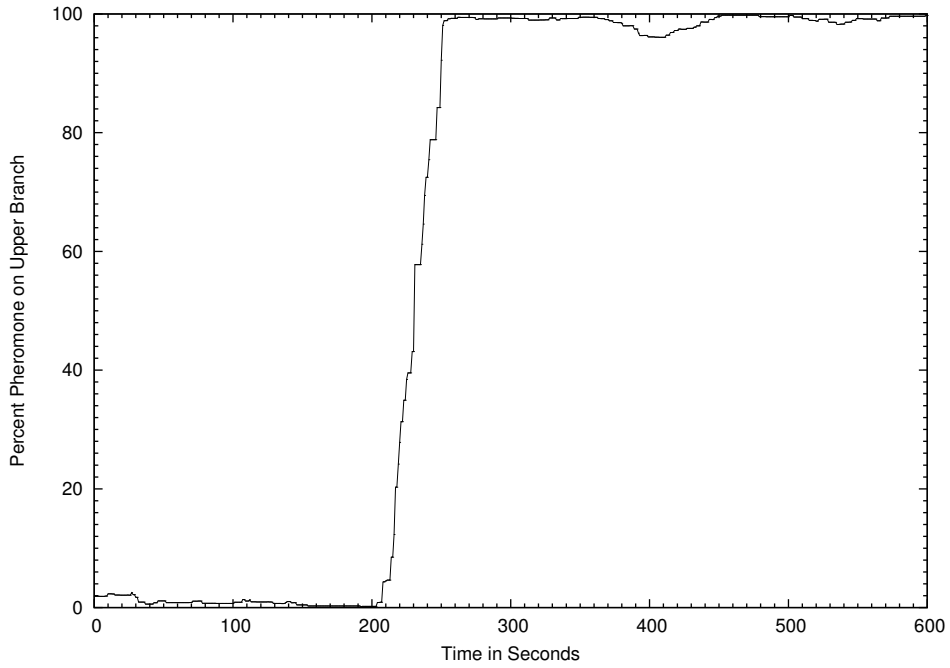

Figure 6.3: Proportion of pheromone on the upper branch according to the route routing table with respect to time.

chosen to be similar to the analytical model update function so as to harness the knowledge unearthed in the previous chapter.

A key difference between these plots and those of Chapter 5, is that these curves are not nearly as smooth as the previous ones. This of course, is to be expected since the plots here represent data collected from a single simulation run. The curves in Chapter 5 are plots of time varying probabilities, and as such, also have the interpretation of being the expected values or averages of certain events. Here, especially in the case of the plot corresponding to the next-hop routing table, we see traces of the ant algorithm working over time in the form of the small peaks and valleys that exist in the curve. At certain times we even see small sojourns away from what might be considered the average value in a given range. These deviations, which are particularily present in the curve corresponding to the next-hop routing table, are a result of an aggressive update function seizing temporarily on a perceived trend. Of course a properly defined update mechanism should also always quickly return to the average value once evidence suggests a change is not necessary. This behaviour is nicely depicted here. The only sojourn that is ultimately realized in terms of a change to a new average value is the one initiated at the switch over time.

While it is interesting to see this level of detail, from here on we will discuss only plots that are actually pointwise averages over several simulation runs. It is important to produce such averaged results for at least 2 main reasons. First, we need to ensure that the results are typical, not rare events that occur so infrequently as to be useless as a guide to our understanding of the routing framework. Technically speaking, Figures 6.2 and 6.3 only provide evidence that for the single associated random seed used in the simulator we arrive at such results. But we want to speak more generally.

If the average of several simulation runs yield similar results we can be confident in what the measures of the ant algorithm are telling us. Second, we anticipate that the averaged plots will be smoother. This is an important fact given that our discussions of the various results hinge on comparing plots, observing trends and noting equilibrium values attained. Smoother curves make it much easier to make these types of arguments.

From this point forward, we will always consider plots that incorporate results from 100 simulation runs. This number has been selected because it is large enough to address the 2 main reasons for looking at averages, as given above, and simultaneously not too large, such that the computational time requirement of the overall simulation is still reasonable. To produce an averaged measure from 100 unique runs, the simulator simply calculates the mean value of the each of the 100 data points at each time value considered.

Utilizing this approach, we replot the pointwise normalized pheromone level average of 100 runs for both the next-hop routing table and the route routing table in Figure 6.4. This time, the 2 measures are easily plotted on the same axes without confusion.

Each curve is much smoother than before, making it much easier to spot features such as limiting values and the steepness of slopes. Moreover, plotted together now, it is even easier to compare the two curves.

We see that the switch over occurs more swiftly in the next-hop routing table. Also, it is quite clear that the original pheromone level in the next-hop routing table is greater than the corresponding level in the route routing table. Similarly, the next-hop routing table equalizes at a point significantly lower than that of the route routing table. These features are all desirable and by design. They are strong indications that

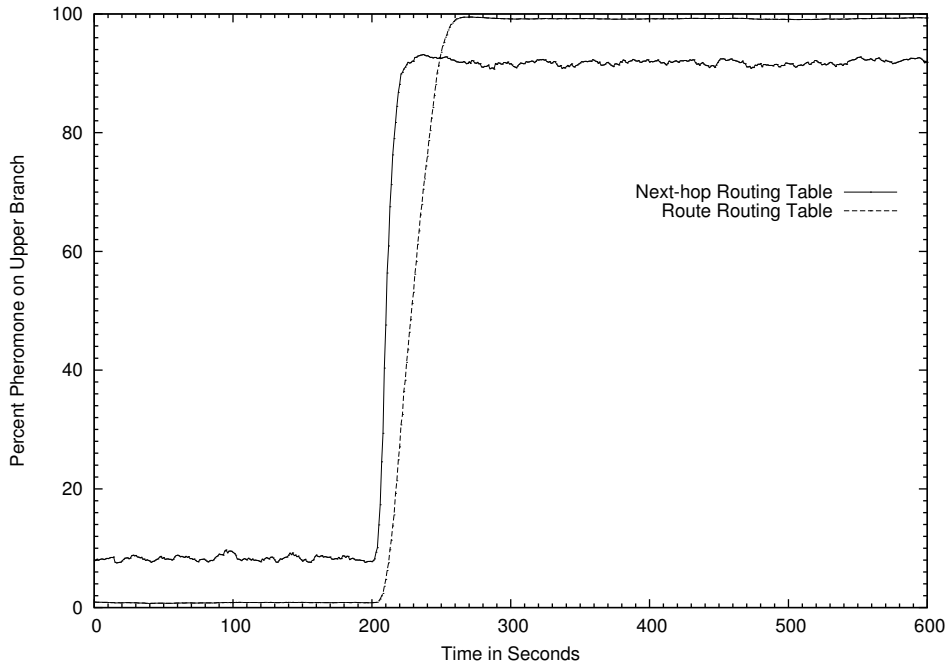

Figure 6.4: Proportion of pheromone on the upper branch according to both routing tables over time. Averaged results for 100 simulations.

decoupling, facilitated by the introduction of the database at node 0, is successful. We can maintain 2 separate and distinct pheromone tables via individually tailored update functions. We can set the update function parameters, based on our study of the embedded urn model, to on one hand result in an aggressive switch over rate for the case of the next-hop routing table, and on the other hand insist the adoption of a new route according to the route routing table is more conservative. Furthermore, we can insist that the point at which the curves begin to level off occurs sooner in the case of the next-hop routing table. We want the normalized pheromone level in the route routing table to rise a slower rate, yet continue to grow for a longer time and ultimately achieves a higher equilibrium value. This is controlled primarily because entries in the next-hop routing table are subject to a prescribed minimum pheromone level threshold. To produce the results seen so far we use a value of 0.005. The notion of decoupling allows us to explicitly bound the the pheromone levels in the next-hop routing table away from the extremes, while simultaneously allowing the pheromone levels in the route routing table to make a closer approach. Through experimentation with the simulator we have also deduced that taking $a_1 > a_0$ can be effective in driving the route routing table levels to more extreme limits, provided we simultaneously use $M_0 < M_1$ to ensure that the relative rates of pheromone buildup are as desired.

If we increase M_0 or decrease a_0 we can force the next-hop routing table to evolve more sluggishly. If we choose to do either of these things however, it is wise to simultaneously increase the next-hop routing table minimum pheromone value to maintain the same ceiling and floor normalized pheromone values that we had before. This adjustment is necessary since the output of the next-hop routing table update

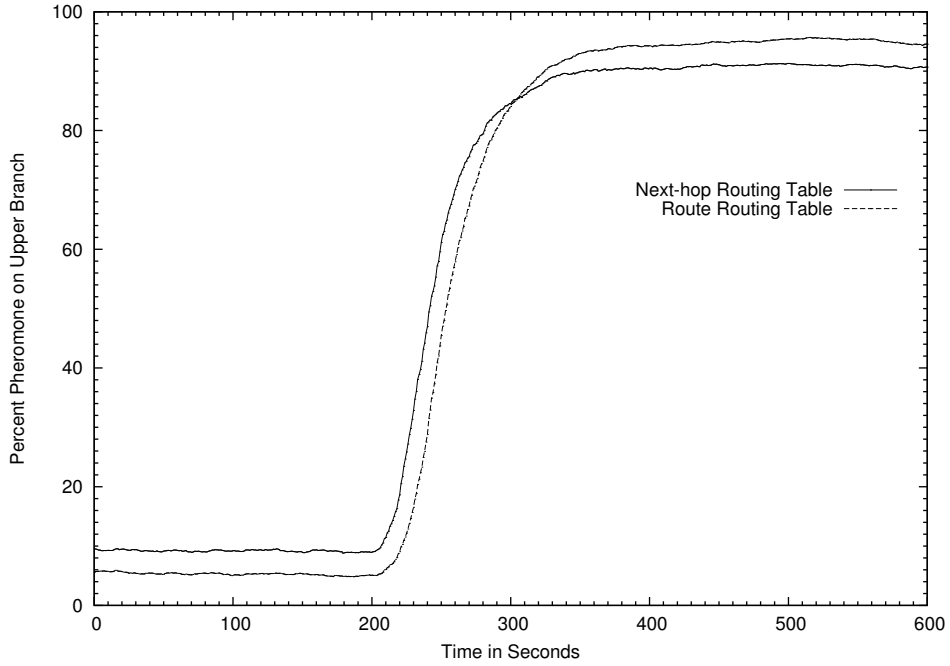

Figure 6.5: Proportion of pheromone on the upper branch according to both tables over time and using a lower a_0 and higher M_0 parameter settings. Pointwise averaged results of 100 simulations.

function will be greater if either the number of items in its sum increases or negative exponential parameter is reduced in magnitude.

In Figure 6.5 we have used a combination $a_0 = 200$ and $M_0 = 20$. We also have taken $a_1 = 200$, since experimentally we have determined it is not wise to have the parameter for the route routing table significantly more sensitive than the corresponding parameter for the next-hop routing table, given the other parameter settings. To balance these changes we take a next-hop routing table minimum pheromone level of 0.2. The overall outcome of these modifications shows both routing tables adapting at a significantly reduced rate. This end result is easily explained by the effects of the changes individually. First, as we know from our mathematical modelling, decreasing

the parameter a_0 makes the next-hop routing table update function less responsive. We are partly seeing this effect here. Second, we recall our intentions for the parameter M_0. Larger values force the update functions to take into account a larger window of feedback. Even as indications point to an underlying traffic load change, the update function hesitates until the evidence is overwhelming. Hence we see that both a_0 and M_0 are vital parameters that have the power to keep the evolution of the next-hop routing table one step ahead at all times.

To get a better idea of the effect of a_0 by itself, we keep $M_0 = 20$ but return a_0 from 200 to 400. We simultaneously set $a_1 = 800$ as before, and adjust the minimum pheromone level to 0.025 so that the pheromone levels are bounded away from the 0% and 100% by approximately 10%, as always. The result of this experiment is shown in Figure 6.6.

Indeed, the rate of increase of the normalized pheromone level in the next-hop routing table associated with the upper branch is increased from the situation in Figure 6.5. This increase, which must be due to the higher setting of a_0, confirms that the potential of this parameter as deduced in our analytic modelling can be realized in an actual ant routing algorithm.

We notice that the results in Figure 6.6, in terms of a preferred situation in which the rise of the next-hop routing table curve significantly outpaces that of the route routing table curve, still fall short of our original plot in Figure 6.4. This suggests that M_0 is an important piece of the puzzle. Adjusting M_0 is not the only way to regain the aggressiveness of the next-hop routing table update functions though. From our analytic modelling we know that a similar result can be achieved through a suitable combination of b_0 and c_0 to initiate downplaying with respect to time. In

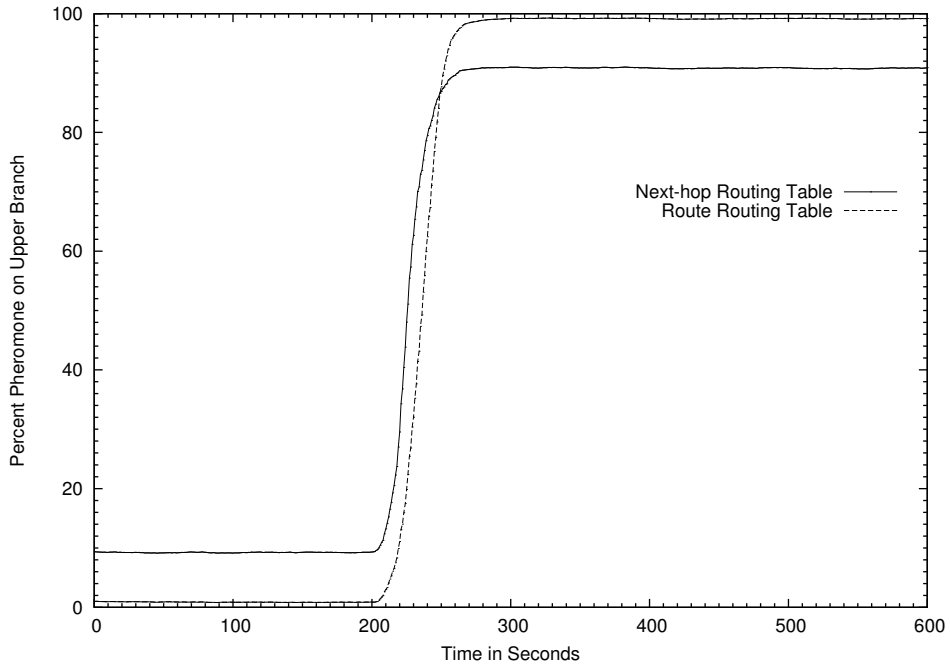

Figure 6.6: Proportion of pheromone on the upper branch according to both tables with respect to time for original a_0 setting and larger M_0 setting. 100 simulations averaged together.

fact, the method ultimately works by faking the same effect as simply reducing M_0. That is, we know that using $M_0 = 5$ allows the next-hop routing table to be more agile than the route routing table by taking into account only 5 pieces of information out of a possible 40 in the database. But another way to effectively focus only on a smaller subset of the information is to increase either b_0 or c_0. We studied this notion of downplaying feedback information based on its age in Chapter 5. In the earlier model we only have a single parameter to adjust. In this context, it is as if we set $b_0 = c_0$. For simplicity of argument here, we will only look at this case. Until now both b_0 and c_0 have been set to 0, but next we test the effect of $b_0 = c_0 = 0.2$. We do so with $M_0 = 20$ so that any increase in the rate of pheromone uptake can be attributed to downplaying with respect to the age of feedback.

As usual in Figure 6.7 we have been careful to also tweak the next-hop routing table minimum absolute pheromone level to ensure typical upper and lower limits of the curve. In this case the original threshold value of 0.005 works well. The outcome confirms that downplaying the pheromone contribution with respect to age is also an effective alternative way to make the rate of pheromone increase more aggressive in our implementation. In fact, the results depicted in Figure 6.7 are essentially identical to those of Figure 6.4. This proves the framework to be highly effective and flexible in the way that we can exert control over the underlying system.

Our ant routing algorithm appears to be making appropriate decisions. We need to check, however, that the routing policy is really doing its job in terms of enabling efficient movement of data throughout the network. We already know that thanks to data packets selecting entire routes from an appropriate route routing table we can guarantee that data is not sent in loops. Hence, the worst that can happen

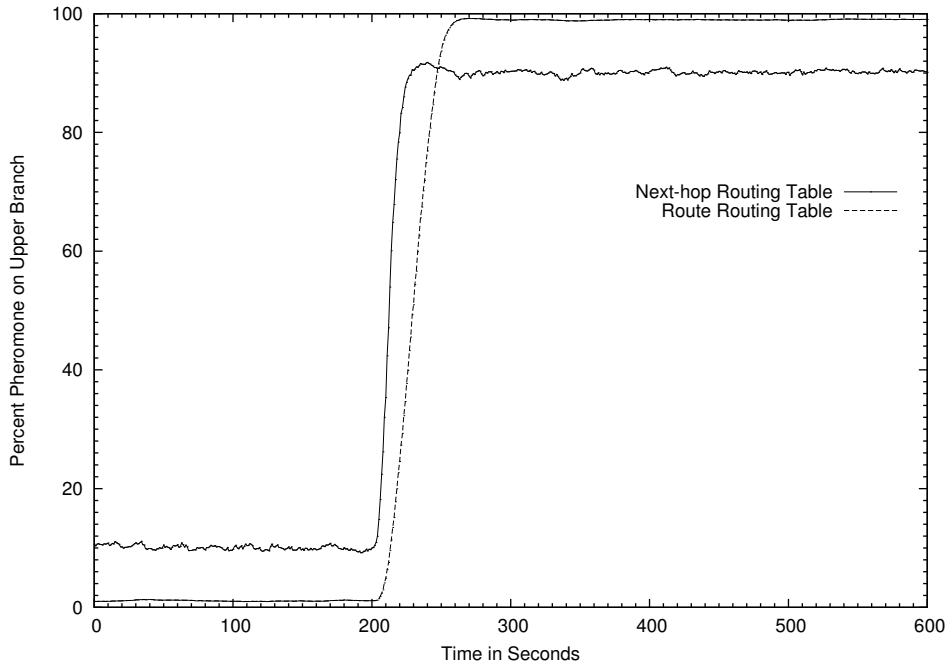

Figure 6.7: Proportion of pheromone on the upper branch according to both the next-hop routing table and route routing table over time with increased memory size M_0 and included downplaying with age. Also averaged over 100 simulations.

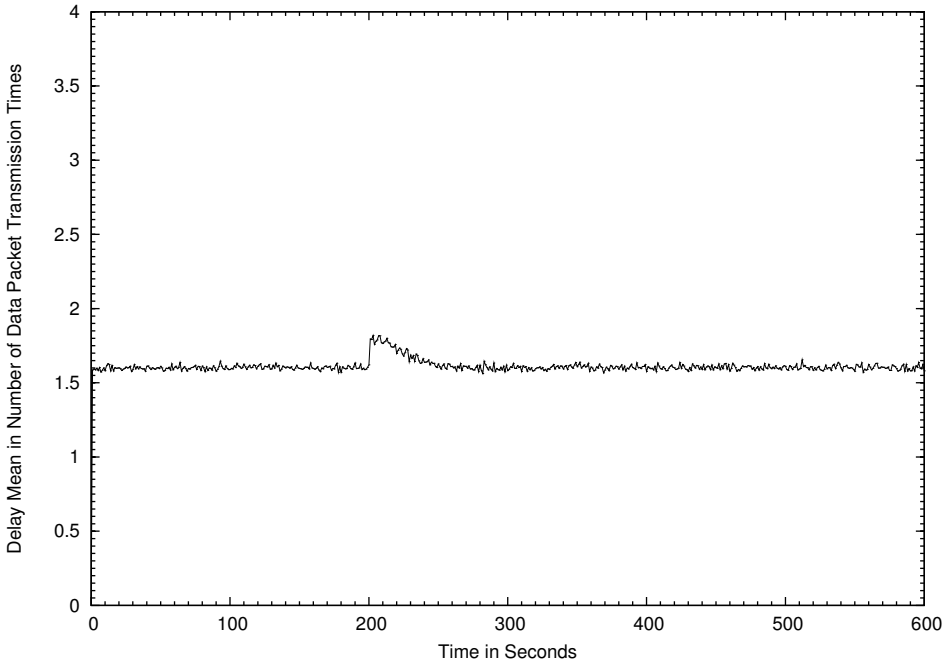

Figure 6.8: Data packet end-to-end delay mean versus time for 100 simulations.

is that a data packet takes too long to reach its destination. More concretely, to judge the experience of data packets in the network, we now turn our attention to data packet end-to-end delay. Between performance data collection times we monitor all data packets transmitted anywhere in the network, together with the total time from their creation at their source node to their eventual arrival at their intended destination node. We plot both end-to-end delay mean and variance values for each interval against time. This yields an approximately instantaneous view of network level delay.

Figure 6.8 shows the data packet end-to-end delay mean, averaged over 100 simulations. We have already seen the simulator parameters used here. We once again take our original settings of $M_0 = 5$, $a_0 = 400$, $a_1 = 800$ and $b_0 = b_1 = c_0 = c_1 = f_1 = 0$.

We note that both before and shortly after the load switch the end-to-end delay mean remains essentially constant at approximately 1.6 data packet transmission times. If the mean delay values were changing sporadically we might infer that the routing algorithm was not performing very well. Stable values indirectly support the belief that the ant routing implementation is behaving as desired. More importantly perhaps, we next concentrate our attention on the time period immediately following the switch over. It is expected that data packets experience larger delays during times of routing table adaptation, so there is no problem with the peak that appears as soon as the second loading scheme comes into effect. It is important however, to evaluate the peak in terms of its severity and total duration. In both aspects the ant based routing algorithm performs well here. The maximum end-to-end delay mean only increases by roughly 1/2 a data packet transmission time at the peak of the change over. Moreover, the measured values are only elevated for a time less than it takes for the route routing table to complete its transition from route 0-2-3 to route 0-1-3. These results suggest that our implementation retains the desirable adaptive load balancing characteristic.

To be absolutely sure that the data packet end-to-end delays are not fluctuating drastically, we plot the associated end-to-end variance in Figure 6.9.

As required, outside the intense period of change the variability of the end-to-end delay estimates is low. Even during the peak of the loading scheme transition the variance is reasonable.

In this example we want to also evaluate the framework in another dimension. The described implementation proposes regular ants that perform both traditional forward and backward functions simultaneously. A potential drawback of this modification

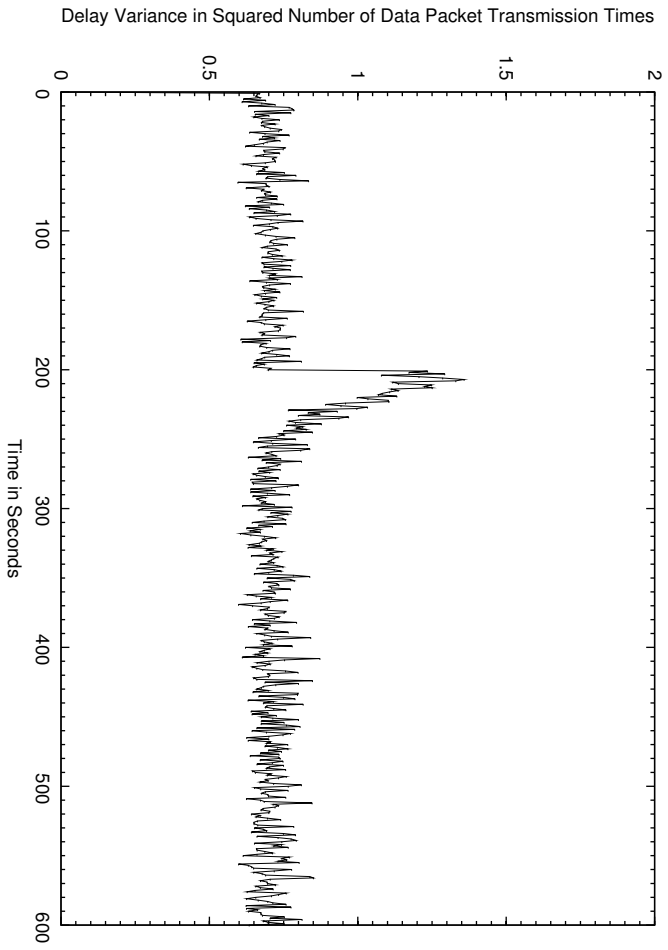

Figure 6.9: Data packet end-to-end delay variance versus time for 100 simulations.

is that routing information that needs to be shuttled around the network may not arrive in a timely manner. Recall that to guard against this pitfall, a provision has been made to include timers that trigger the dispatch of special backward ants before undelivered feedback information becomes irrelevant. So there exists a trade-off between the additional overhead of special backward ants and the latency in delivering routing information. We claim that we can control the trade-off by adjusting the time after which the special backward ants are dispatched.

Although it has not been stated explicitly, up to this point the maximum time before a backward ant is dispatched for any piece of feedback has been taken to be 2 seconds. It turns out that this is a safe setting that ensures a reasonable trade-off between the number of backward ants needed to deliver routing information and the time before this information is used in the form of updates, the latter being measured in ant wave inter-event times here. To illustrate control over the trade-off though, in Figures 6.10 and 6.11 we plot the percentage of backward ants required versus time and update latency versus time respectively for several different values of the simulator feedback information expiration time. All other simulator parameters remain unchanged from the previous experiment, and as always, we plot pointwise averages over 100 simulations in both cases. Over the range of expiration times that we consider, specifically values of 1.0, 2.0 and 3.0, there is no appreciable change in the measures we have already explored at the single expiration time of 2.0. We do not explicitly confirm this fact here to save space, but it is important to note that control of the efficient use of ants does not come at the expense of performance in other areas.

In terms of the proportion of backward ants being dispatched, 2 things are immediately clear from Figure 6.10. First, as evidenced by the nearly constant values of the

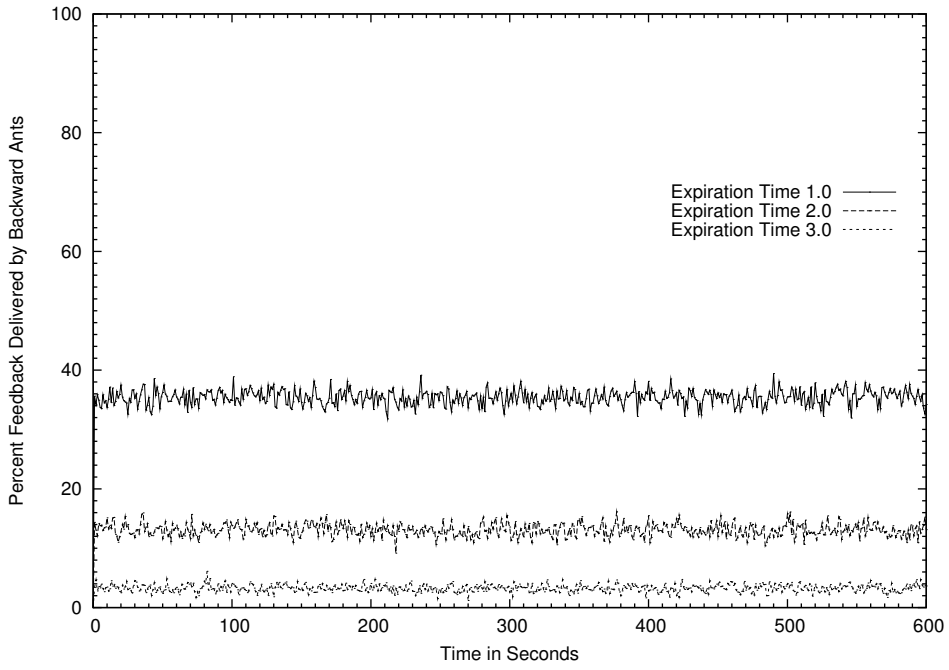

Figure 6.10: Proportion of backward ants delivering feedback against time for various feedback expiration times. Each curve averaged over 100 simulations.

curves at each parameter setting, the average number of backward ants flowing in the network appears very stable over the period presented. This is desirable behaviour in itself since it implies that the ant routing algorithm maintains a relatively constant need for backward agents. The alternative, in which the number of backward ants oscillates with time, could induce oscillations in the flow of data traffic as well, leading to a volatile situation. Second, the various curves point to a clear trend in the proportion of backward ants dispatched in view of the expiration timer setting. As the expiration time is decreased, more and more backward agents are being utilized. This makes perfect sense given the definition of the feedback expiration time. In fact, in the limit as the expiration time approaches 0, we also know that 100% of the feedback is delivered by ants performing backward function only. Finally, we see that feedback expiration times near 2 seconds lead to approximately 15% of the feedback delivered by backward ants. This, we argue, is only a marginal increase in the routing algorithm overhead.

Considering the associated update latency results in Figure 6.11, we can now complete the argument outlining the control of efficient feedback dissemination. In this final plot we once again see 2 main properties. As before, we first note that the the latencies for each of the expiration times are essentially constant. For the same reasoning we suggested earlier, this is ideal. A stable, constant stream of feedback is intuitively preferred over one that surges sporadically. Also as in the previous plot, the second thing we notice is that there is a clear trend in latency versus the feedback expiration time parameter. This time however, we note the opposite trend. As the expiration time decreases, so too does the time between feedback generation and its eventual use in the form of updates to pheromone tables. There are limits on the update

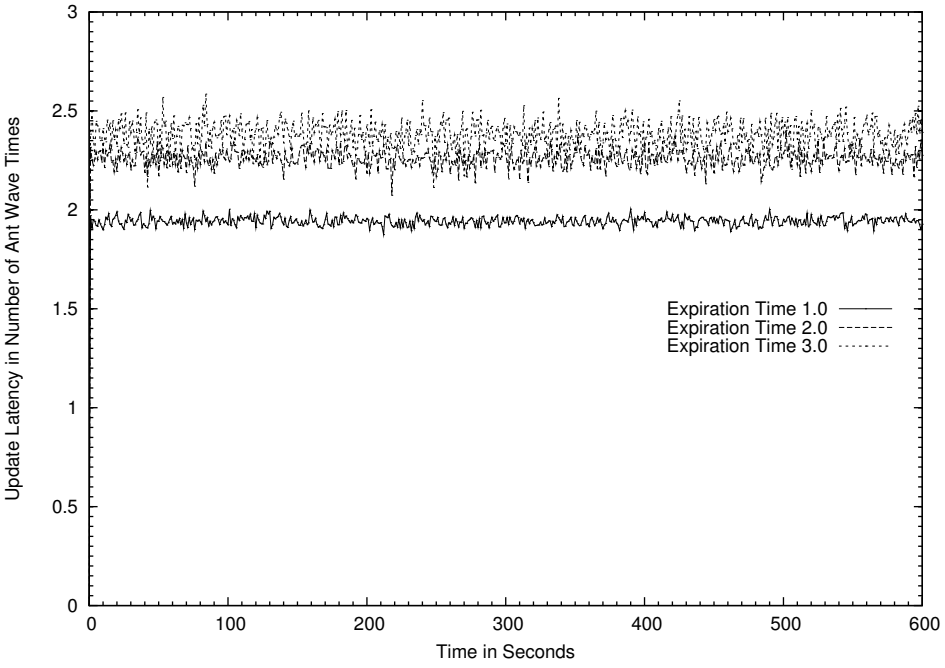

Figure 6.11: Update latency in ant-wave inter event times with respect to time for various feedback expiration times. Each curve averaged over 100 simulations.

latency values. The smallest value is 1 ant-wave inter event time. This is a side effect of the way in which the simulator is coded. The expiration of feedback timers is only checked at the beginning of each new ant wave. Therefore, the earliest time information collected in a given wave will be delivered is the time between consecutive ant waves. With this in mind though, the latencies displayed in Figure 6.11 are all reasonable. In particular, when the feedback information expiry time is close to 2, routing information is passed along in this network in approximately 2.3 ant waves inter event times. Meanwhile, only a modest number of backward ants are required.

6.3.2 Network Level Control Simulation

The previous example investigates a single choice point. Its primary purpose is to solidify our understanding of the proposed ant routing algorithm framework in an environment that is easy to visualize. This example is a natural extension where we verify that the level of control that exists in a simple setting does in fact scale to larger networks. Here we consider Canada's 20 node national testbed network, CA*net 4. The structure of this network for our purposes is pictured in Figure 6.12.

While we typically prefer to number the nodes in the network for use in the simulator, as we have done in the diagram of CA*net 4, it is worth noting some of the major urban centres represented. Table 6.1 lists the cities corresponding to each node.

It is easy to see that the building blocks of the larger network we consider here are rings, which we expect to behave locally the same as the 4 node ring we studied earlier. For this reason, we use essentially the same parameter settings as in the previous example. For instance, we will take $M = 40$, $M_0 = 5$ and $M_1 = 40$ to

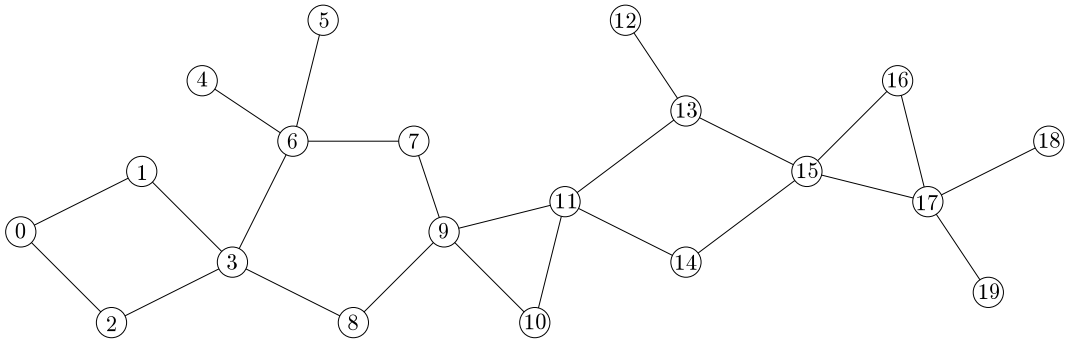

Figure 6.12: CA*net 4 topology.

Node Number	City Name
0	Victoria
1	Vancouver
2	Seattle
3	Calgary
4	Whitehorse
5	Yellowknife
6	Edmonton
7	Saskatoon
8	Regina
9	Winnipeg
10	Chicago
11	Toronto
12	Iqaluit
13	Ottawa
14	New York
15	Montreal
16	Fredericton
17	Halifax
18	Charlottetown
19	St. John's

Table 6.1: List of city names corresponding to the numerically labelled nodes of CA*net 4.

parameterize databases and update functions. Moreover, we will assume that data packet transmission time is once again 0.005 seconds, that the feedback information expiration time is 2 seconds and we will take performance measurements at 1 second intervals. We also take $a_0 = 400$ and $a_1 = 800$ as before, and we continue to ignore certain factors, for simplicity, by setting $b_0 = b_1 = c_0 = c_1 = f_1 = 0$.

In terms of the traffic loading, we once again specify simple schemes that allow us to anticipate the desired response of the ant routing algorithm. Essentially we imagine the loading is uniform except at a few key points, where we alter it to study specific choice points. We want to focus on node 0, node 3 and node 11. We assume majority of the nodes transmit data to every other node at rate 1 packet per second. Before the switch over node 1, node 8 and node 13 transmit at rate 120 packets per second. After the switch, these nodes resume sending at rate 1 packet per second and in their place, nodes 2, 7 and 14 take over sending at rate 120 packets per second. These settings are sufficient to induce congestion at particular points in the network causing certain paths to be either preferred strongly or not.

Given the larger size of CA*net 4 however, we do need to adjust some parameter settings. This time we take the ant wave inter event time to be 0.2 seconds to ensure there is a decent amount of feedback being generated per unit time. A smaller value is needed because each node dispatches an ant to only one random destination each ant wave, and the number of possible destination nodes has increased. Notice that a recalculation of ant versus data packet demands on the network suggests that on average for this network example we expect 120 ants/second, again assuming 20% of the time backward ants are used, compared with 737 data packets/second. This implies a ratio of ant objects to data packets objects of roughly 14%, an increase

from the much smaller 4 node network considered before. However, as we argued in that earlier example, in reality ants require perhaps only 10% of the capacity of data packets. Therefore, a better estimate of the overhead required for ants in this larger network is approximately 1.6%. With heaver data traffic, this value might even be decreased. Finally, we use a next-hop routing table minimum pheromone value of 0.005, since once again it is sufficient to ensure a reasonable amount of exploration of alternate routes at all times.

We consider the same basic measures here that we discussed earlier in the 4 node ring. The goal is simply to verify the desirable properties that we discovered earlier apply in this setting as well. This time we will jump immediately to plots of pointwise averages. We will consider the average of 100 simulation runs throughout the discussion here.

As in the first example, we start out by considering the normalized pheromone levels in both the next-hop routing table and the route routing table. Because of the loading scheme chosen this time though, there are 3 different areas of the network worth probing. From Figure 6.12 we see that they correspond to 3 different ring structures, which we will say each contain 2 alternate paths designated the upper and lower branches of the ring. We then look at the normalized pheromone levels on the upper branches over time, using the same logic as for the 4 node ring example.

The first ring we consider is labelled identically to our previous example. That is, ring 0-1-3-2-0. As before we take the upper branch to be 0-1-3, making 0-2-3 the alternative. Because of the loading schemes selected, we expect this ring sub structure to behave in much the same way as well. That is, since node 1 effectively congests the upper branch before the load switch we expect traffic from node 0 to node 3 to initially take the lower branch. Later, as node 2 congests the lower branch and there

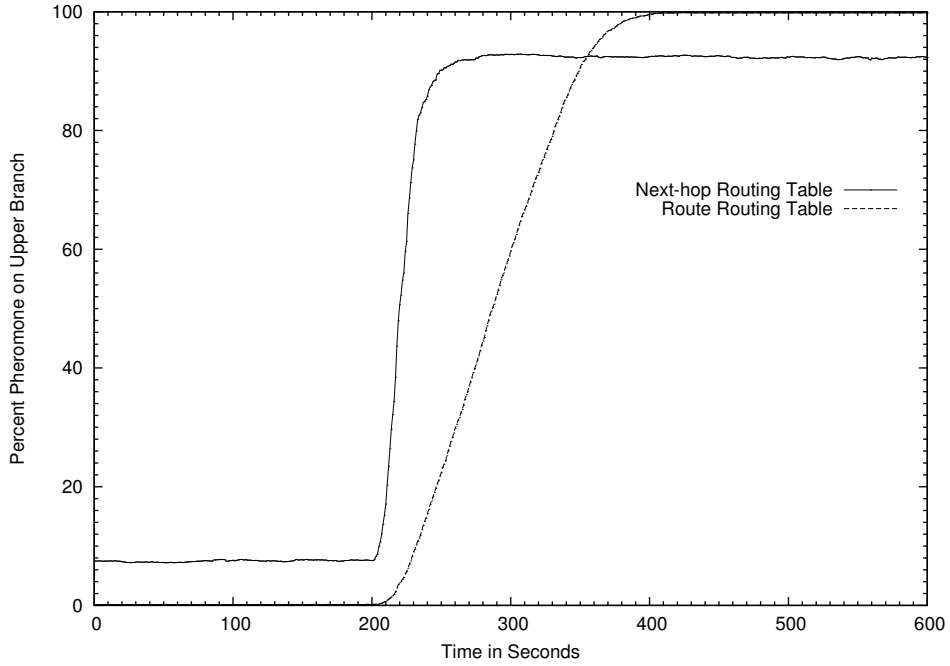

Figure 6.13: Proportion of pheromone on route 0-1-3 according to both the next-hop routing table and route routing table with respect to time. 100 simulations incorporated in the average.

is relief on the upper branch, we anticipate a dramatic rebalancing as the routing algorithm eventually prefers the upper branch for traffic from node 0 to node 3. In fact, this theory is nicely confirmed in Figure 6.13.

Once again, because of appropriate update function settings, both curves are seen to make the right choices, with the curve associated with the next-hop routing table leading the way. Moreover, as always the next-hop routing table function starts from a higher relative position and equalizes at a lower pheromone level, exactly as required. The only real appreciable difference between this plot and that of Figure 6.4 is that there is a slightly larger lag in the evolution of the tables here. This can

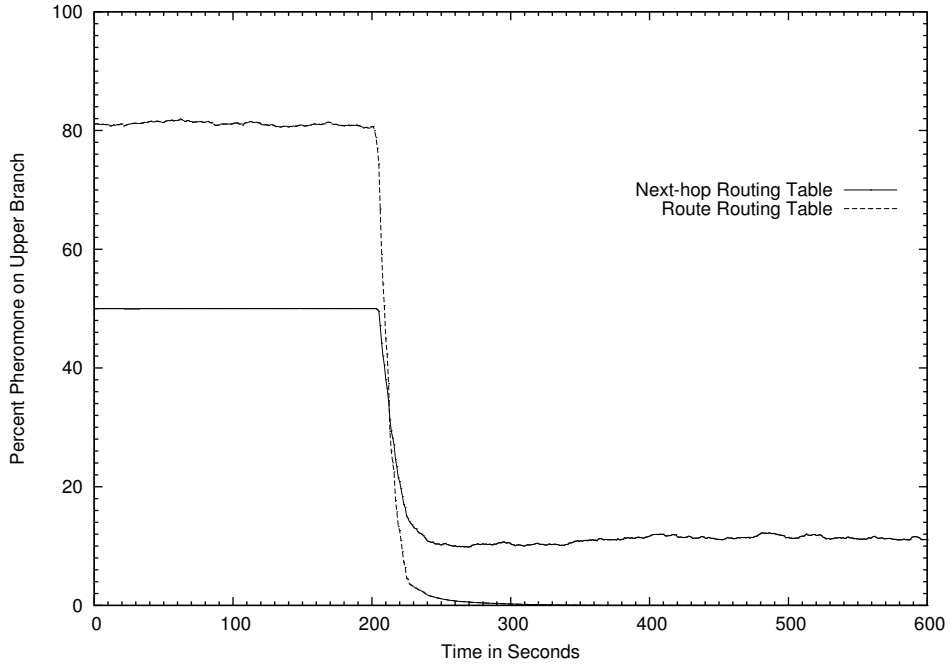

Figure 6.14: Proportion of pheromone on route 3-6-7-9 according to both the next-hop routing table and route routing table with respect to time. 100 simulations incorporated in the average.

easily be attributed to the increased workload of the ants given the increased number of possible destination from every source. Even though the time between ant waves is decreased, there is still slightly less information about a given route per unit time.

The second structure to consider brings a little variety to an otherwise repetitive discussion. This time we consider the ring 3-6-7-9-8-3, which has 5 nodes instead of 4. We imagine that 3-6-7-9 constitutes the upper branch in this case, leaving 3-8-9 to fill the role of the alternate, lower branch. Another key difference this time is that the loading is such that, if anything, the preferred branches before and after the load switch will be reversed.

Looking at Figure 6.14 we see these differences. We note that each of the curves begin higher and end lower, opposite to any curves we have seen thus far. This is attributed to the reversed loading. The more interesting feature in this case however, is that the higher values of each curve are not as large as before. This is especially true of the next-hop routing table related curve. The reason for this is the difference in route lengths of the upper and lower branch in this sub ring, and it highlights the fact that total path length is dynamic. In the period before the switch over the algorithm must decide between using a route that has 3 hops as opposed to 2, or accept a higher than average congestion level. From the point of view of the next-hop routing table updates the two choices are essentially the same, and they must both be monitored equally for any change of state. At the same time, since the route routing table update function is more sensitive, it still favours the upper branch before the switch, although to a much lesser extreme. One final thing to note regarding these two plots is that while we typically prefer to observe the next-hop routing table serving as the guide, leading the way in advance of any route routing table evolution, this does not appear to be the case here. The explanation for this fact is simply that sometimes when the choice is quite clear, both update functions are able to discern this fact essentially in parallel. This is the situation here of course. After the load switch the preferred path is the one that is both shorter and less congested.

The third ring we consider lies deeply within the CA*net 4. It is yet another 4 node ring, specifically consisting of nodes 11-13-15-14-11, with equal length upper and lower branches. We take the upper route to be 11-13-15, while we take the lower route in this case to be 11-14-15. The interesting point this time is that being essentially in the middle of network, we expect this ring experience the effect of a good deal of

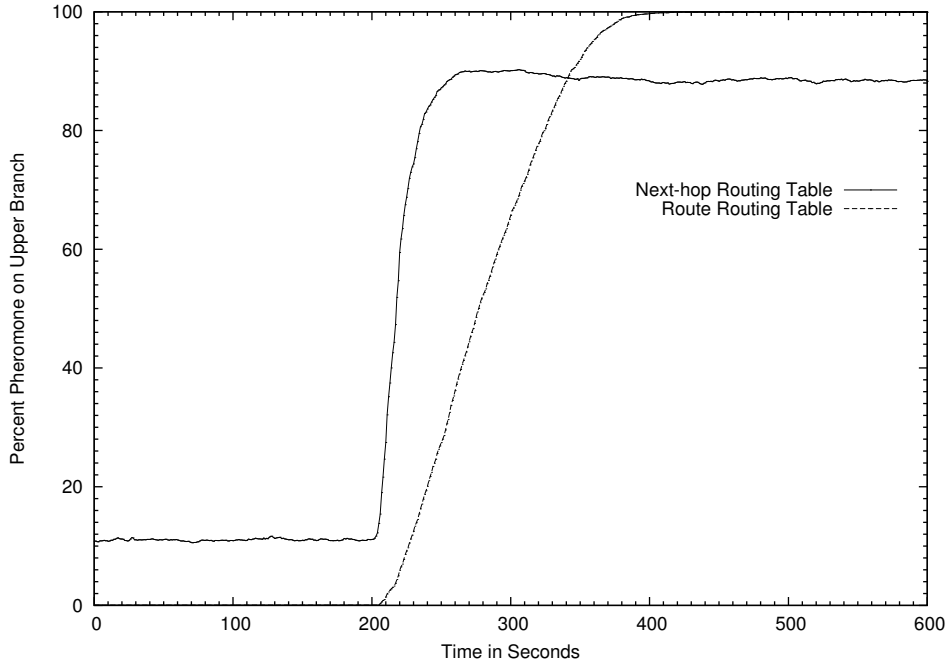

Figure 6.15: Proportion of pheromone on route 11-13-15 according to both the next-hop routing table and route routing table with respect to time. 100 simulations incorporated in the average.

external traffic. Still, a simple analysis of the average loading suggests that the upper branch should be highly undesireable before the switch over, and highly desirable after the switch over. We want to confirm then, that in this more complex local environment the ant routing scheme is still functioning correctly. Figure 6.15, which plots the normalized pheromone level associated with the upper branch for both the next-hop routing table used by the ants and the route routing table used by data packets, confirms that the algorithm is still effective in this situation.

In fact, Figure 6.15 is barely distinguishable from either Figure 6.13 or even Figure 6.4. This suggests more than just the fact that the ant routing algorithm is appropriate for

various networks. It indicates that a single set of parameters can effectively manage a variety local conditions simultaneously. This is an important observation in terms of real-world implementation. It hints that optimizations for local conditions may not be necessary, although they can easily be added to the framework. That is, there is no requirement for each node to run with identical update functions, but it would be convenient, especially given the typically decentralized or hands off benefit of ant algorithms.

So far, there seems to be no surprises regarding the pheromone tables we have interrogated. It only remains to show that a global measure of performance does not contradict our belief that the ant routing framework is working well. For this, we return to monitoring end-to-end delay experienced by the data packets. We once again look at both the end-to-end delay mean and end-to-end delay variance of every packet delivered up to each data collection event. The results are given in Figures 6.16 and 6.17 respectively.

The end-to-end delay mean versus time plot is similar to the analogous result for the 4 node ring example given in Figure 6.8. The mean is once again stable, albeit at a larger value of 4.1 data packet transmission times, except for a short time immediately following the loading switch over. The fact that the mean is larger is easily understood once we realize that the measure is accounting for traffic on all routes, and on average, the routes are longer here. The fact that there is a peak is not worrisome. As the change over occurs, we expect there to be a limited buildup of data packets at various output queues. The main thing to verify is that the buildup is temporary. Indeed, in this case the increased congestion is only for a duration approximately the same length of time as it takes for the route routing table to evolve, which we note on

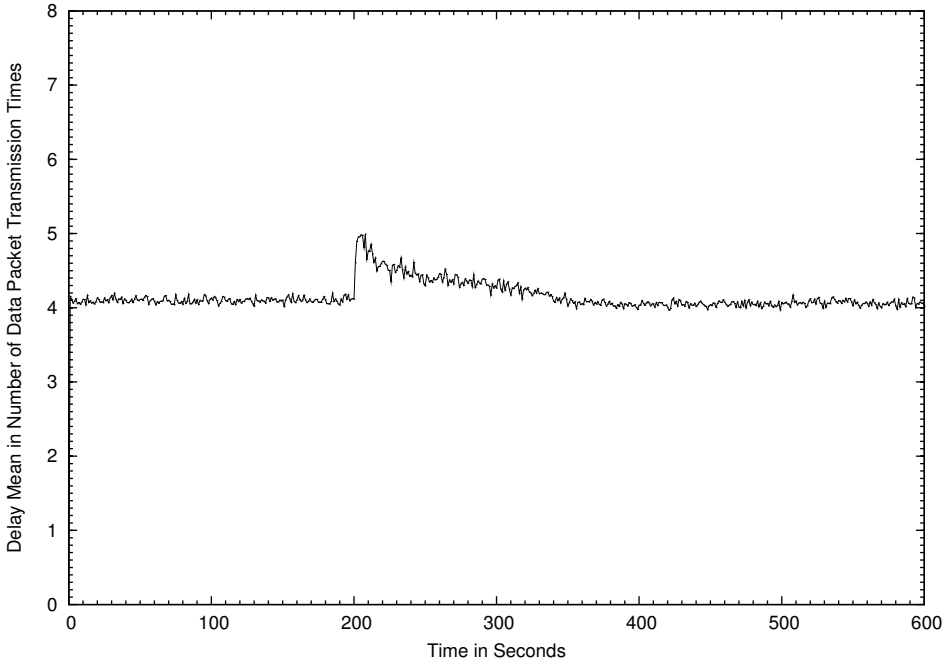

Figure 6.16: Data packet end-to-end delay mean for CA*net 4 versus time. 100 simulations used to create the pointwise average.

average from Figures 6.13, 6.14 and 6.15.

The variance measure, even more directly perhaps, supports the theory that the traffic load is managed well at all times. It too shows an increase in the transient period following the switch from the first loading scheme to the second, but overall, it is stable at a reasonable value. Our mean values in Figure 6.16 are averaged over the number of successful data packet deliveries that occur since the last performance measure was taken. The variance values presented here too are only over the window of time between data collections. We anticipate there to be some variance. Partly because of changing delays on the same routes, but mostly because of the naturally varied end-to-end delay values from different routes with different route lengths. Taking all this into mind, the variance pictured here is reasonable and shows no signs of undesirable behaviour such as large sporadic fluctuations.

Finally, before we leave this larger network example, we want to consider the usage of ants. Earlier, in a smaller network, we were able to control the trade-off between using more agents, which is a greater routing policy overhead, and the result of using more agents, which reduces the time between generating feedback and utilizing it. More importantly perhaps, we determined that our ant routing framework works well with a minimum number of backward ants required for operation. Here too, we will see that these comments are valid.

In terms of the number of backward ants being dispatched for a given feedback expiration time, Figure 6.18 yields similar results to those in Figure 6.10 from the stand alone 4 node ring example.

Once again we see that the average ratio of the number of backward ants moving feedback to the total number of ants moving feedback is essentially constant at each

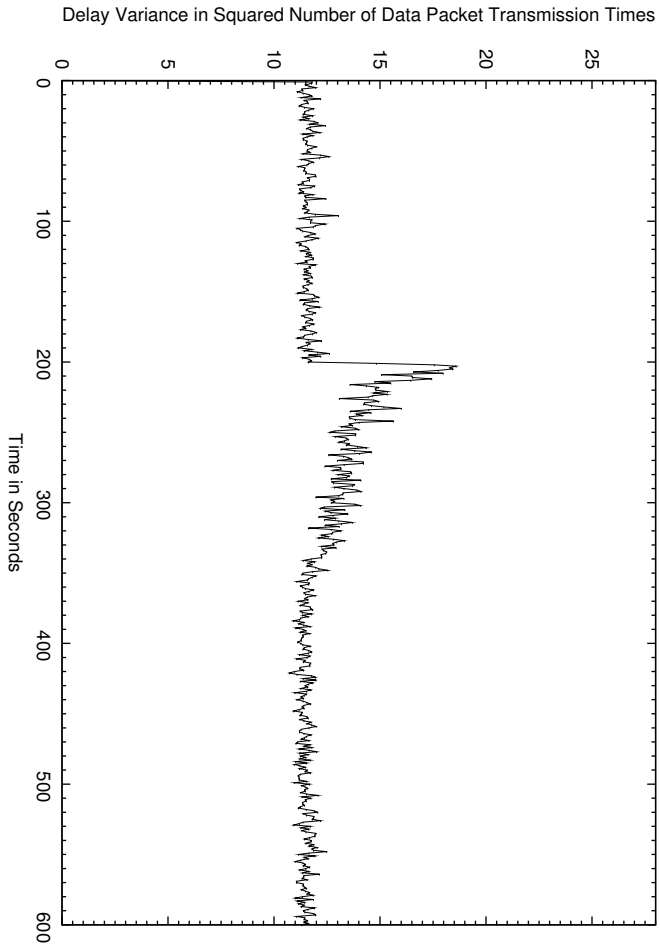

Figure 6.17: Data packet end-to-end delay variance for CA*net 4 versus time. 100 simulations used to create the pointwise average.

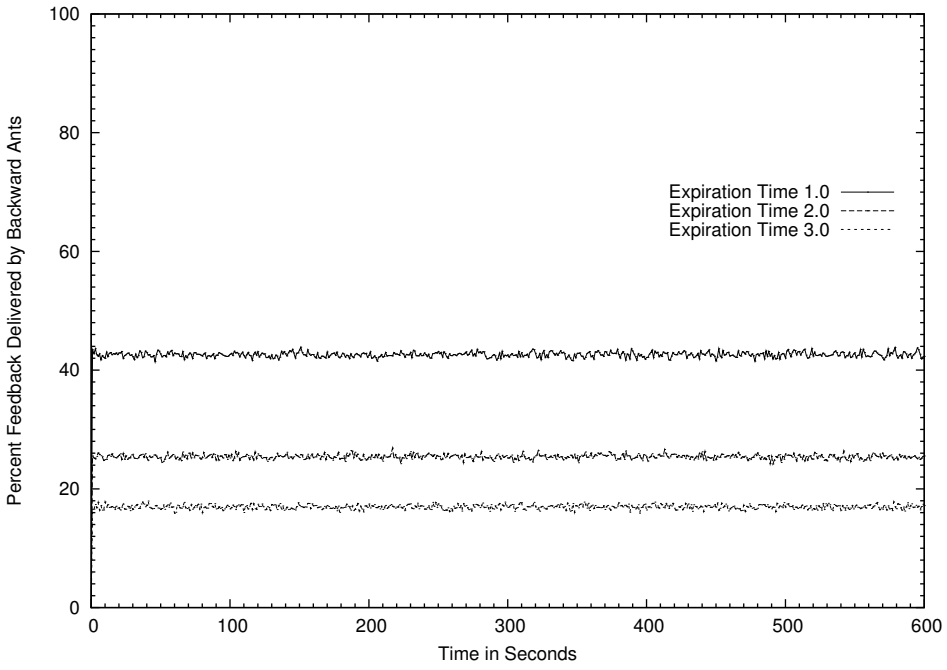

Figure 6.18: Proportion of backward ants delivering feedback for CA*net 4 versus time for various feedback expiration times. Results for 100 simulations together.

value of 1.0, 2.0 and 3.0 for the feedback expiration time. For the same reasoning we expressed in the previous example, this is the preferred situation. We want the ant routing implementation to maintain a minimal but steady flow of backward ants. Additionally, as the expiration time is decreased, we once again observe the proportion of backward ants dispatched to carry feedback increases. The reason for this is intuitively evident. The main difference in this larger example is that the percentage of backward ants for each of the expiration times considered are larger than in the 4 node ring. For instance, when the expiration timer is around 2 seconds, the percentage of backward ants is approximately 25%. This is mainly a result of a minimal number of ants flowing in this example. If we increase the number of ants that traverse the network we increase the chance that a given piece of feedback will be delivered before the expiration time triggers the dispatch of a backward ant.

In order to fully appreciate the trade-off between algorithm overhead and efficiency, we must consider what we have termed the update latency. This measure, which is plotted in Figure 6.19 for CA*net 4, gives us a picture of the real-time latencies associated with each piece of feedback. Between data collections, the elapsed time from creation to final delivery for each piece of feedback dropped off by any type of ant is monitored. The average is plotted versus time.

Figure 6.19 shows us 3 such curves, one corresponding to each feedback expiry time of 1.0, 2.0 and 3.0. As always, each of the curves are fairly smooth and level. This in itself suggests the ant algorithm is performing in such a way that it is stable with resect to delivering updates. Moreover, we have already encountered the trend that emerges in this plot. As the feedback expiration time decreases, Figure 6.11 demonstrates a similar decrease in update latency. The underlying explanation is

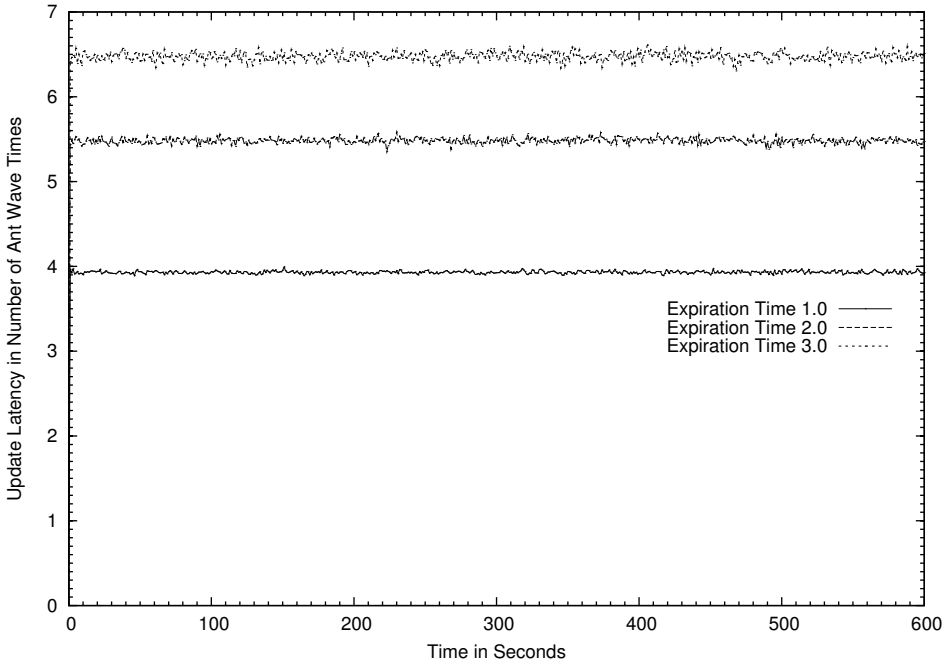

Figure 6.19: Update latency in ant-wave inter event times for CA*net 4 versus time for various feedback expiration times. 100 simulation runs averaged.

more or less obvious since reducing the expiration time is essentially equivalent to saying we are less willing to wait for feedback to make updates. As noted earlier, if the expiration time approaches 0, the feedback will never be more than 1 ant wave old. Of course in a real implementation, where the movement of ants is not centralized as it is in the simulator, as the expiration time vanishes the feedback becomes practically instantaneous. The only real difference we may note here is that the update latencies are significantly larger than in the 4 node ring example. In fact, they are essentially only larger in the number of ant waves, not time. Remembering our approximately 2 fold reduction of the time between ant waves, which we consciously modified due to the larger number of nodes in this network, helps justify the increase in ant wave times before feedback is delivered and updates are applied. As an example, for an expiration time of 2 seconds it takes roughly 5.5 ant wave inter event times before feedback is applied. But this only represents 1.1 seconds when ant waves are 0.2 seconds apart. For the 4 node ring, with ant waves that are 0.4 seconds apart, the analogous length of time is 2.3 ant waves or 0.9 seconds. Hence, the ant routing algorithm scales quite well with respect to this measure. In fact, a larger network with diverse routes may actually out perform a smaller one in regard to update latency since there is more chance for regular ants fulfilling primarily forward duties to disseminate pending feedback in the process. At any rate, taken together now, the results in Figure 6.18 and Figure 6.19 strongly suggest that we will not encounter problems delivering routing information in a timely manner, and using minimal overhead, in any real world implementation of this ant routing framework.

6.4 Discussion

This chapter attempts to pull together all our knowledge of ant algorithms gained via analytic modelling and make use of it to propose a novel ant routing algorithm framework. To do so, we have introduced three complementary data structures at every router, each of which provides a different type of control. First, a database is proposed to retain recent routing information to allow for more flexible table updates that can respond to observed traffic trends. This allows for greater control over how conservatively or aggressively the tables track different types of load change. Complementary to this is the second data structure, a separate route routing table for data packets, which officially decouples control over data packet behaviour from the control over the behaviour of ants. Having separate routing tables truly allows freedom to optimize the objectives of ant routing and data packet routing in conjunction. Moreover, we insist that the entries of the route routing table specify complete routes, thus guaranteeing loop-free routing for data packets. The third data structure added to the framework is a deliverables array used to hold routing information waiting to be carried by ants to other nodes. Ideally, we do not want to generate backward ants to deliver this information, rather the goal is to have already flowing ants transmit the majority of feedback in the network. The deliverables array provides flexibility in controlling a trade-off between making the most out of existing ants, which perform both forward and backward functions, and delay in routing information delivery.

The simulation results of the framework are very encouraging. In both small and large networks it is possible to demonstrate the idea of decoupling that is enabled by having a common database of routing feedback information with two separate routing tables built from it. Viewing both the next-hop pheromone table and the route pheromone

table at several decision points and under varying conditions, we have shown that tailor made update functions can harness the gathered feedback and use it for specific purposes. Furthermore, the same update procedures maintain reasonable end-to-end delays for the data traffic. In times of change, all measures indicate a tendency to quickly return to stable equilibrium values. Finally, our experimentation also reveals quite favourable results in terms of the redefined use of routing agents. It appears that giving each ant the ability to deliver feedback can transmit routing information throughout the network with acceptable latency using far fewer agents.

Additionally, we have seen that the techniques proposed in this chapter can be implemented without much trouble. A prime example is the database. Our software simulation works well storing no more than 40 pieces of data corresponding to each destination. Even if each piece is several bytes and there are a hundred or more destinations, for example, this still only represents a small database by today's standards. Moreover, it should not be hard to calculate functions of this small set of values when performing updates. Dedicated processors should be able to manage all required calculations for a large volume of both ants and data packets. Finally, even our simplistic process of sending ants in synchronized waves has suggested that the notion of having ants perform both forward and backward functions simultaneously is practical and efficient. We expect any real implementation, which would most likely dispatch a greater number of ants asynchronously, to perform just as well or better than what we have presented above.

A real world ant routing implementation would necessarily require more attention to detail than can possibly be given here. For instance, concrete methods for tuning the parameters of any chosen update functions would have to be established. Ideally, the

mechanisms might even be automated. The main purpose of this chapter then, with respect to implementation, is more proof of concept. We note that our ant algorithm framework maintains all of the highly desireable characteristics of traditional ant algorithms, such as robustness, adaptability and scalability, thanks to simple, local and decentralized interactions between agents. More importantly, we show that with the help of additional data structures at each node and a small overhead of ants flowing between them, perhaps on a dedicated channel, a real world network could maintain very effective routing control using our framework.

Chapter 7

Conclusions and Future Work

In this final chapter we make some concluding remarks on this portion of the work as a whole. Of course research is ongoing, and we will continue developing ideas related to those presented above. In the following we also discuss some of the topics we intend to study further.

7.1 Comprehensive Discussion

Our curiosity in ant algorithms has led to exciting mathematical modelling opportunities. In turn, this analytic investigation has led to new probabilistic results and an improved design for a potentially large class of ant algorithms. This is an ideal situation with respect to mathematics motivated by a desire to better understand a real world problem.

Initially we cultivate a simple finite urn into a suitable model for static link cost ant algorithms. Almost immediately though, it becomes obvious the representation is a

much more general purpose probability tool. Indeed, we confirm this by using it to model the error processes in a example on binary channels. Even without this bonus however, our functional weight finite urn validates itself by quickly yielding worthwhile measures of steady state performance for the intended ant routing application.

We have also taken the time to explore the limiting behaviour of a generalized finite urn in the linear form irreducible case. We note that there are actually 2 possible limits to consider. The functional weight finite urn is parametrized by a fixed memory size, and it is a process over time. We can imagine the limiting scaled weight of each of the colour types in the urn in both of these dimensions. It is somewhat intriguing however, that the order in which we do so is crucial. On one hand, it is clear that letting the memory go to infinity first effectively produces an infinite urn. Hence, considering the time limit after the memory limit is not new, and the two urns are equivalent in this case. On the other hand, we can calculate the time limit first, which yields the stationary composition of an urn with finite memory, and then the limit as the memory size increases. It is tempting to assume that the result is the same as the previous ordering, but it turns out that this is not true. The corresponding limiting scaled mean vectors for the finite and infinite urns are still the same in this case, but the respective limiting scaled covariance matrices are not. Moreover, whereas the infinite urn limiting analysis breaks into 3 separate classes, in the finite urn there is only 1. Empirically it seems that the limiting distribution of a sequence of scaled finite urn compositions is always normal for the single class.

For all its generality, the functional weight finite urn is still not complex enough to be used to evaluate ant algorithms where dynamic link costs must also be tracked in order to follow the urn evolution. Since one objective of this work is to analyze ant-based

routing in communication networks, a more advanced model is required. Indeed, part of this research extends the generalized finite urn model. The result is termed an embedded functional weight finite urn because of the way it implicitly models an urn under the effects of random external processes that could represent other urns working simultaneously. That is, the embedded finite urn can be thought of as a model for a specific finite urn among a network of similar urns. By design, this is exactly the kind of situation we require for ant routing. We use this model, with its external process component, to represent a single node in an ant algorithm in the presence of many other nodes. The analysis allows us to quickly test pheromone table update functions, and otherwise build our knowledge of the relationship between potential tunable ant routing parameters and their effect on the outcome. Later, we exploit this awareness in designing an actual ant-based routing algorithm implementation.

We undertake a mathematical approach to analyzing ant algorithms to put them on a solid footing, and ideally to understand them well enough to propose enhancements. In fact, in the end we suggest modifications that essentially amount to an entirely different ant-based paradigm. By incorporating a database, we remove the burden of pheromone tables to act as a collective memory. The history of pertinent feedback is housed in the database, leaving the pheromone tables free to evolve as dramatically as deemed necessary. Moreover, depending on the problem to be solved, we propose using more than one pheromone table built from the database. In our network routing application for instance, we use one pheromone routing table for ants and a separate pheromone routing table for data packets. This decoupling alleviates a common challenge in current ant routing algorithms. We can allow the ants to aggressively

monitor the network and adapt quickly when necessary, yet still handle data in prudent and responsible way. Our new architecture even attempts to use every ant as efficiently as possible. In the case of ant routing, it is also easy to include provisions such that data packets never get caught on looping routes. Generally speaking, the proposed framework should permit increased control over ant algorithms for a variety of problems.

7.2 Continuing Research

It is extremely rare for any piece of research to be so self contained that it is considered absolutely complete. This work is no exception. In terms of both the probabilistic models and the ant algorithm framework, there are ample opportunities for future work.

The functional weight finite urn is a promising mathematical model. Infinite urns have been used successfully in many ways, and we expect finite urns to be just as versatile. Basically, for each case in which an infinite urn may be used, there is potentially a corresponding finite version to consider as well, while in certain situations, just as in the modelling of ant algorithms, the finiteness may be crucial. Hence, one type of extended research is to consider other possible applications of finite urns.

More importantly perhaps, we wish to continue investigating the limiting normality hypothesis for a sequence of stationary scaled finite urn compositions. So far, it seems to be a difficult result to prove, but perhaps further insight into this conjecture will come in the future.

Regarding our ant algorithm framework, there are lots of interesting questions to be

tackled. The simulation study of ant routing could be expanded to consider various update function forms for both the next-hop routing table used by ants, and the route routing table used by data packets. Part of this task might include the development of methods to automatically tune required update function parameters. Also, different network topologies could be tested. Furthermore, once the proposed ant routing algorithm is optimized, it would be nice to compare it with other routing algorithms. Finally, it would be interesting to explore the potential benefit of the new framework to ant algorithm applications other than network routing.

The field of biologically inspired algorithms is still young. Just as this project develops interesting mathematics by looking at the current state of ant-based procedures, there will be further opportunities for this sort of analytic research in the future. Mathematics will continue to play an important role in shaping this elegant solution method.

Bibliography

[1] F. Alajaji and T. Fuja, "A communication channel modeled on contagion", *IEEE Transactions on Information Theory*, 40(6):2035-2041, November, 1994.

[2] K. Athreya and S. Karlin, "Embedding of Urn Schemes Into Continuous Time Markov Branching Processes and Related Limit Theorems", *Ann. Math. Stat.*, 39:1801-1817, 1968.

[3] K. Athreya and P. Ney, *Branching Processes*, Springer, 1972.

[4] N. Bean and A. Costa, "An Analytic Modelling Approach for Network Routing Algorithms That Use Ant-like Mobile Agents", to appear in *Computer Networks*, 2005.

[5] D. Bertsekas and R. Gallager, *Data Networks,* 2^{nd} *Ed.*, Prentice Hall, 1992.

[6] E. Bonabeau, F. Henaux, S. Guérin, D. Snyers, P. Kuntz and G. Theraulaz, "Routing in Telecommunications Networks with Smart Ant-like Agents", *Proceedings of the Second International Workshop on Intelligent Agents for Telecommunication Applications*, 60-71, 1998.

[7] E. Bonabeau, M. Dorigo and G. Theraulaz, *From Natural to Artificial Swarm Intelligence*, New York: Oxford University Press, 1999.

[8] E. Bonabeau, M. Dorigo and G. Theraulaz, "Inspiration for Optimization from Social Insect Behaviour", *Nature*, 406:39-42, 2000.

[9] L. Bianchi, L. Gambardella and M. Dorigo, "An Ant Colony Optimization Approach to the Probabilistic Traveling Salesman Problem", *Proceedings of the Seventh International Conference on Parallel Problem Solving from Nature*, 883-892, 2002.

[10] A. Colorni, M. Dorigo and V. Maniezzo, "Distributed Optimization by Ant Colonies", *Proceedings of European Conference on Artificial Life*, 134-142, 1991.

[11] A. Colorni, M. Dorigo and V. Maniezzo, "An Investigation of some Properties of an Ant Algorithm", *Proceedings of the Parallel Problem Solving from Nature Conference*, 509-520, 1992.

[12] D. Corne, M. Dorigo and F. Glover, *New Ideas in Optimization*, McGraw-Hill, 1999.

[13] A. Costa, "Analytic Modelling of Agent-Based Network Routing Algorithms", Ph.D. Thesis, The University of Adelaide, 2002.

[14] T. Cover and J. Thomas, *Elements of Information Theory*, New York: Wiley, 1991.

[15] J. Deneubourg, S. Aron, S. Goss and J. Pasteels, "The Self-Organizing Exploratory Pattern of the Argentine Ant", *Journal of Insect Behavior*, 3(2):159-168, 1990.

[16] G. Di Caro and M. Dorigo, "AntNet: Distributed Stigmergetic Control for Communications Networks", *Journal of Artificial Intelligence Research*, 9:317-365, 1998.

[17] M. Dorigo, "Optimization, Learning and Natural Algorithms", Ph.D. Thesis, Politecnico di Milano, Italy, 1992.

[18] M. Dorigo, V. Maniezzo and A. Colorni, "The Ant System: Optimization by a Colony of Cooperating Agents", *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 26(1):1-13, 1996.

[19] M. Dorigo, G. Di Caro and L. Gambardella, "Ant Algorithms for Discrete Optimization", *Artificial Life*, 5(2):137-172, 1999.

[20] M. Dorigo and T. Stützle, *Ant Colony Optimization*, MIT Press, 2004.

[21] W. Dyczka, "On the Multidimensional Pólya Distribution", *Ann. Soc. Math. Polonae, Ser. I: Comment. Math.*, 17:43-63, 1973.

[22] F. Eggenberger and G. Pólya, "Über die Statistik verketteter Vorgänge", Zeitschrift fur Angewandte Mathemathik und Mechanik, 3:279-289, 1923.

[23] E. Elliot, "Estimates of error rates for codes on burst-noise channels," *Bell Systems Technical Journal*, 42:1977-1997, September 1963.

[24] S. Franklin and A. Graesser, "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents", *Proceeding of the Third International Workshop on Agent Theories, Architectures, and Languages*, 21-35, 1996.

[25] B. Fritchman, "A binary channel characterization using partitioned Markov chains," *IEEE Transactions on Information Theory*, 13(2):221-227, April 1967.

[26] L. Gambardella and M. Dorigo, "Ant-Q: A Reinforcement Learning Approach to the Traveling Salesman Problem", *Proceedings of the Twelfth International Conference on Machine Learning*, 252-260, 1995.

[27] E. Gilbert, "Capacity of a burst-noise channel," *Bell Systems Technical Journal*, 39:1253-1266, September 1960.

[28] S. Goss, S. Aron, J. Deneubourg and J. Pasteels, "Self-Organized Shortcuts in the Argentine Ant", *Naturwissenschaften*, 76(12):579-581, 1989.

[29] P. Grassé, "La Reconstruction du nid et les coordinations interindividuelles. La théorie de la stigmergie", *Insectes Sociaux*, 6:41-84.

[30] W. Gutjahr, "A Graph-based Ant System and its Convergence", *Future Generation Computer Systems*, 16(8):873-888, 2000.

[31] W. Gutjahr, "ACO Algorithms with Guaranteed Convergence to the Optimal Solution", *Information Processing Letters*, 82(3):145-153, 2002.

[32] M. Heusse, S. D. Snyers, Guérin and P. Kuntz, "Adaptive Agent-Driven Routing and Load Balancing in Communication Networks", *Advances in Complex Systems*, 1(2):237-254, 1998.

[33] J. Hoffmeyer, "The Swarming Body", *Proceedings of the Fifth Congress of the International Association for Semiotic Studies*, 937-940, 1997.

[34] S. Janson, "Functional Limit Theorems for Multityple Branching Processes and Generalized Pólya Urns", *Stochastic Processes and Their Applications*, 110:177-245, 2004.

[35] S. Janson, "Limit Theorems for Triangular Urn Schemes", to appear in *Probability Theory and Related Fields*, 2005.

[36] N. Johnson and S. Kotz, *Urn Models and Their Application,* John Wiley & Sons, 1977.

[37] D. Merkle and M. Middendorf, "Modeling the Dynamics of Ant Colony Optimization", *Evolutionary Computation*, 10(3):235-262, 2002.

[38] K. Oida and A. Kataoka, "Lock-Free AntNets and their Adaptability Evaluations", *IEICE Trans. on Communications*, J82-B(7):1309-1319, 1999.

[39] S. Ross, *Introduction to Probability Models,* 8^{th} *Ed.*, Academic Press, 2003.

[40] R. Schoonderwoerd, O. Holland, J. Bruten and L. Rothkrantz, "Ant-Based Load Balancing in Telecommunications Networks", *Adaptive Behaviour*, 5(2):169-207, 1996.

[41] E. Seneta, *Non-negative Matrices and Markov Chains,* 2^{nd} *Ed.*, Springer-Verlag, 1981.

[42] C. Shannon, "A Mathematical Theory of Communication", *Bell Systems Technical Journal*, 27:379-423,623-656, 1948.

[43] A. Silva, A. Romão, D. Deugo and M. Mira Da Silva, "Towards a Reference Model for Surveying Mobile Agent Systems", *Autonomous Agents and Multi-Agent Systems*, 4(3):187-231, 2001.

[44] T. Stützle and H. Hoos, "MAX-MIN Ant System", *Future Generation Computer Systems*, 16(8):889-914, 2000.

[45] T. Stützle and M. Dorigo, "A Short Convergence Proof for a Class of Ant Colony Optimization Algorithms", *IEEE Transactions on Evolutionary Computation*, 6(4):358-365, 2002.

[46] G. Varela and M. Sinclair, "Ant Colony Optimization for Virtual-Wavelength-Path Routing and Wavelength Allocation", *Proceedings of Congress on Evolutionary Computation*, 1809-1816, 1999.

[47] S. Verdú and T. Han, "A General Formula for Channel Capacity", *IEEE Transactions on Information Theory*, 40(4):1147-1157, 1994.

[48] J. Watmough and L. Edelstein-Keshet, "Modelling the Formation of Trail Networks by Foraging Ants", *Journal of Theoretical Biology*, 176(3):357-371, 1995.

[49] N. Weber, "A Martingale Approach to Central Limit Theorems for Exchangeable Random Variables", *Journal of Applied Probability*, 17:662-673, 1980.

[50] T. White, "Swarm Intelligence and Problem Solving in Telecommunications", *Canadian Artificial Intelligence magazine*, 14-17, 1997.

[51] L. Zhong, F. Alajaji and G. Takahara, "A Binary Communication Channel with Memory Based on a Finite Queue," submitted to *IEEE Transactions on Information Theory*, 2004.

[52] L. Zhong, F. Alajaji and G. Takahara, "An Approximation of the Gilbert-Elliot Channel Via a Queue-Based Channel Model", *Proceedings of the 2004 IEEE International Symposium on Information Theory*, 63, 2004.

[53] L. Zhong, "A Binary Burst-Noise Communication Channel Modeled by a Finite Queue: Information Theoretic Properties and Applications to Wireless Communications", Ph.D. Thesis, Queen's University, 2005.

[54] L. Zhong, F. Alajaji and G. Takahara, "A Queue-Based Model for Wireless Rayleigh Fading Channels with Memory", to appear in *Proceedings of the IEEE* 62^{nd} *Semiannual Vehicular Technology Conference*, 2005.

Appendix A

Additional Theorems

This appendix contains theory developed over the course of the research that lacks a good fit elsewhere in this document. Although not needed directly for the presentation of main results earlier, the derivations given here can yield additional insight into limiting relationships.

Theorem A.1 *Weak Assertion*

$$\begin{aligned}\lim_{M\to\infty}\left(p'_{null|ij}(M)-p'_{null|i}(M)\right) &= \lim_{M\to\infty}\left(p'_{null|j}(M)-p'_{null}(M)\right)\\ &= 0.\end{aligned}$$

Proof *First note that* $q'_{ij}(M)=p'_{ij}(M)+p'_{null,ij}(M)$*, where* $p'_{null,ij}$ *is the probability of slot 0 containing colour* i*, slot 1 containing colour* j *and drawing the null ball.*

Next, consider

$$
\begin{aligned}
& p'_{null|ij}(M) - p'_{null|ik}(M) \\
&= \frac{1}{q'_{ij}(M)} p'_{null,ij}(M) - \frac{1}{q'_{ik}(M)} p'_{null,ik}(M) \\
&= \frac{1}{q'_{ij}(M)} (q'_{ij}(M) - p'_{ij}(M)) - \frac{1}{q'_{ik}(M)} (q'_{ik}(M) - p'_{ik}(M)) \\
&= \frac{1}{q'_{ik}(M)} p'_{ik}(M) - \frac{1}{q'_{ij}(M)} p'_{ij}(M) \\
&= \frac{1}{q'_{ik}(M)} \sum_{x:x_0=k} \frac{w_i(x)}{w'} \pi_{X',x}(M) - \frac{1}{q'_{ij}(M)} \sum_{x:x_0=j} \frac{w_i(x)}{w'} \pi_{X',x}(M) \\
&= \frac{1}{q'_{ik}(M)} \sum_{s:s_k \geq 1} \frac{w_i(x^*(s))}{w'} \binom{M-1}{s_0, \ldots, s_k - 1, \ldots, s_{C-1}} \pi'(x^*(s)) \\
&\quad - \frac{1}{q'_{ij}(M)} \sum_{s:s_j \geq 1} \frac{w_i(x^*(s))}{w'} \binom{M-1}{s_0, \ldots, s_j - 1, \ldots, s_{C-1}} \pi'(x^*(s)) \\
&= \frac{1}{M q'_{ik}(M)} \sum_{s} \frac{w_i(x^*(s)) s_k}{w'} \binom{M}{s_0, \ldots, s_{C-1}} \pi'(x^*(s)) \\
&\quad - \frac{1}{M q'_{ij}(M)} \sum_{s} \frac{w_i(x^*(s)) s_j}{w'} \binom{M}{s_0, \ldots, s_{C-1}} \pi'(x^*(s)) \\
&= \frac{1}{M q'_{ik}(M)} \sum_{x} \frac{w_i(x) s_k(s)}{w'} \pi_{X',x}(M) - \frac{1}{M q'_{ij}(M)} \sum_{x} \frac{w_i(x) s_j(x)}{w'} \pi_{X',x}(M) \\
&= \frac{1}{M w' q'_{ik}(M)} \sum_{x} \left(\alpha_i + \sum_{h=0}^{C-1} \Delta_{ih} s_h(s) \right) s_k(s) \pi_{X',x}(M) \\
&\quad - \frac{1}{M w' q'_{ij}(M)} \sum_{x} \left(\alpha_i + \sum_{h=0}^{C-1} \Delta_{ih} s_h(s) \right) s_j(x) \pi_{X',x}(M) \\
&= \frac{1}{M w' q'_{ik}(M)} \left(\alpha_i E[N'_k] + \sum_{h=0}^{C-1} \Delta_{ih} E[N'_h N'_k] \right) \\
&\quad - \frac{1}{M w' q'_{ij}(M)} \left(\alpha_i E[S'_j] + \sum_{h=0}^{C-1} \Delta_{ih} E[N'_h S'_j] \right) \\
&= \frac{1}{w' q'_{ik}(M)} \left(\alpha_i E[N'_k]/M + \sum_{h=0}^{C-1} \Delta_{ih} E[N'_h N'_k]/M \right) \\
&\quad - \frac{1}{w' q'_{ij}(M)} \left(\alpha_i E[S'_j]/M + \sum_{h=0}^{C-1} \Delta_{ih} E[N'_h S'_j]/M \right)
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{1}{w' q'_{ik}(M)} \left(\alpha_i E[N'_k]/M + \sum_{h=0}^{C-1} \Delta_{ih} \sigma'_{hk} + \sum_{h=0}^{C-1} \Delta_{ih} E[N'_h] E[N'_k]/M \right) \\
&\quad - \frac{1}{w' q'_{ij}(M)} \left(\alpha_i E[S'_j]/M + \sum_{h=0}^{C-1} \Delta_{ih} \sigma'_{hj} + \sum_{h=0}^{C-1} \Delta_{ih} E[N'_h] E[S'_j]/M \right) \\
&= \frac{1}{w' q'_{ik}(M)} \left(\alpha_i q'_k + \sum_{h=0}^{C-1} \Delta_{ih} \sigma'_{hk} + q'_k \sum_{h=0}^{C-1} \Delta_{ih} E[N'_h] \right) \\
&\quad - \frac{1}{w' q'_{ij}(M)} \left(\alpha_i q'_j + \sum_{h=0}^{C-1} \Delta_{ih} \sigma'_{hj} + q'_j \sum_{h=0}^{C-1} \Delta_{ih} E[N'_h] \right) \\
&= \frac{1}{w' q'_{ik}(M)} \left(\alpha_i q'_k + \sum_{h=0}^{C-1} \Delta_{ih} \sigma'_{hk} \right) - \frac{1}{w' q'_{ij}(M)} \left(\alpha_i q'_j + \sum_{h=0}^{C-1} \Delta_{ih} \sigma'_{hj} \right) \\
&\quad + \left(\frac{q'_k}{w' q'_{ik}(M)} - \frac{q'_j}{w' q'_{ij}(M)} \right) \sum_{h=0}^{C-1} \Delta_{ih} E[N'_h] \\
&= \frac{1}{w' q'_{ik}(M)} \left(\alpha_i q'_k + \sum_{h=0}^{C-1} \Delta_{ih} \sigma'_{hk} \right) - \frac{1}{w' q'_{ij}(M)} \left(\alpha_i q'_j + \sum_{h=0}^{C-1} \Delta_{ih} \sigma'_{hj} \right) \\
&\quad + \frac{(q'_k q'_{ij}(M) - q'_j q'_{ik}(M))}{w' q'_{ik}(M) q'_{ij}(M)} \sum_{h=0}^{C-1} \Delta_{ih} E[N'_h].
\end{aligned}
$$

But

$$
\begin{aligned}
&q'_k q'_{ij}(M) - q'_j q'_{ik}(M) \\
&= q'_k \frac{1}{(M-1)} (\sigma'_{ij} + M q'_i q'_j - q'_i I_{\{i=j\}}) - q'_j \frac{1}{(M-1)} (\sigma'_{ik} + M q'_i q'_k - q'_i I_{\{i=k\}}) \\
&= \frac{1}{(M-1)} (q'_k \sigma'_{ij} - q'_k q'_i I_{\{i=j\}} - q'_j \sigma'_{ik} + q'_j q'_i I_{\{i=k\}}).
\end{aligned}
$$

So we have

$$
\begin{aligned}
&p'_{null|ij}(M) - p'_{null|ik}(M) \\
&= \frac{1}{w' q'_{ik}(M)} \left(\alpha_i q'_k + \sum_{h=0}^{C-1} \Delta_{ih} \sigma'_{hk} \right) - \frac{1}{w' q'_{ij}(M)} \left(\alpha_i q'_j + \sum_{h=0}^{C-1} \Delta_{ih} \sigma'_{hj} \right)
\end{aligned}
$$

$$+\frac{M}{(M-1)w'q'_{ik}(M)q'_{ij}(M)}\sum_{h=0}^{C-1}\Delta_{ih}q'_h\left(q'_k\sigma'_{ij}-q'_kq'_iI_{\{i=j\}}-q'_j\sigma'_{ik}+q'_jq'_iI_{\{i=k\}}\right).$$

Next, we remove the dependence on k by multiplying by $\frac{q'_{ik}(M)}{q'_i(M)}$ and summing over k as in

$$\begin{aligned}
&p'_{null|ij}(M)-p'_{null|i}(M)\\
&= \sum_{k=0}^{C-1}\frac{q'_{ik}(M)}{q'_i(M)}(p'_{null|ij}(M)-p'_{null|ik}(M))\\
&= \sum_{k=0}^{C-1}\frac{1}{w'q'_i(M)}\left(\alpha_iq'_k+\sum_{h=0}^{C-1}\Delta_{ih}\sigma'_{hk}\right)-\sum_{k=0}^{C-1}\frac{q'_{ik}(M)}{q'_i(M)w'q'_{ij}(M)}\left(\alpha_iq'_j+\sum_{h=0}^{C-1}\Delta_{ih}\sigma'_{hj}\right)\\
&\quad+\sum_{k=0}^{C-1}\frac{M}{(M-1)w'q'_i(M)q'_{ij}(M)}\sum_{h=0}^{C-1}\Delta_{ih}q'_h\left(q'_k\sigma'_{ij}-q'_kq'_iI_{\{i=j\}}-q'_j\sigma'_{ik}+q'_jq'_iI_{\{i=k\}}\right)\\
&= \frac{1}{w'q'_i(M)}\alpha_i-\frac{1}{w'q'_{ij}(M)}\left(\alpha_iq'_j+\sum_{h=0}^{C-1}\Delta_{ih}\sigma'_{hj}\right)\\
&\quad+\frac{M}{(M-1)w'q'_i(M)q'_{ij}(M)}\sum_{h=0}^{C-1}\Delta_{ih}q'_h\left(\sigma'_{ij}-q'_iI_{\{i=j\}}+q'_jq'_i\right).
\end{aligned}$$

But if we take $M\to\infty$, we note that $w'\to\infty$, and each term above disappears. That is, we can show that

$$\begin{aligned}
&p'_{null|ij}-p'_{null|i}\\
&= \lim_{M\to\infty}\left[p'_{null|ij}(M)-p'_{null|i}(M)\right]\\
&= \lim_{M\to\infty}\left[\frac{1}{w'q'_i(M)}\alpha_i-\frac{1}{w'q'_{ij}(M)}\left(\alpha_iq'_j+\sum_{h=0}^{C-1}\Delta_{ih}\sigma'_{hj}\right)\right.\\
&\quad\left.+\frac{M}{(M-1)w'q'_i(M)q'_{ij}(M)}\sum_{h=0}^{C-1}\Delta_{ih}q'_h\left(\sigma'_{ij}-q'_iI_{\{i=j\}}+q'_jq'_i\right)\right]\\
&= 0.
\end{aligned}$$

But of course in a similar way we can remove the dependence on i by multiplying by $\frac{q'_{ij}(M)}{q'_j(M)}$ and summing over i to show

$$\begin{aligned} & p'_{null|j} - p'_{null} \\ &= \lim_{M\to\infty}\left[\sum_{i=0}^{C-1}\frac{q'_{ij}(M)}{q'_j(M)}\left(p'_{null|ij}(M) - p'_{null|i}(M)\right)\right] \\ &= \sum_{i=0}^{C-1}\left(\lim_{M\to\infty}\frac{q'_{ij}(M)}{q'_i(M)}\right)(0) \\ &= 0. \end{aligned}$$

□

Corollary A.2 *In fact*

$$\lim_{M\to\infty}(r'_i(M)q_i(M)) = q_i$$

and

$$\lim_{M\to\infty}\left(r'_{ij}(M)q_{ij}(M)\right) = q_{ij}.$$

Proof *Concerning the first statement we have*

$$\begin{aligned} \lim_{M\to\infty}(r'_i(M)q_i(M)) &= \lim_{M\to\infty}\left(\frac{(1-p'_{null}(M))}{(1-p'_{null|i}(M))}q_i(M)\right) \\ &= \frac{(1-p'_{null})}{(1-p'_{null})}\lim_{M\to\infty}q_i(M) \\ &= q_i. \end{aligned}$$

Similarly, for the second statement, since

$$\begin{aligned}\lim_{M\to\infty}\left(p'_{null|ij}(M)-p'_{null}(M)\right) &= \lim_{M\to\infty}\left(p'_{null|ij}(M)-p'_{null|i}+p'_{null|i}-p'_{null}(M)\right)\\ &= 0\end{aligned}$$

we also have

$$\begin{aligned}\lim_{M\to\infty}\left(r'_{ij}(M)q_{ij}(M)\right) &= \lim_{M\to\infty}\left(\frac{(1-p'_{null}(M))}{(1-p'_{null|ij}(M))}q_{ij}(M)\right)\\ &= \frac{(1-p'_{null})}{(1-p'_{null})}\lim_{M\to\infty}q_{ij}(M)\\ &= q_{ij}.\end{aligned}$$

□

Theorem A.3 *Strong Assertion*

The following statements are equivalent:

(i)

$$\begin{aligned}\lim_{M\to\infty}M(p'_{null|ij}(M)-p'_{null|i}(M)) &= \lim_{M\to\infty}M(p'_{null|j}(M)-p'_{null}(M))\\ &= -\frac{1}{q_j\Delta_{MCS}}\sum_{h=0}^{C-1}\Delta_{CS,h}\sigma_{hj}\end{aligned}$$

(ii)

$$\sigma_{ij}=\lim_{M\to\infty}\sigma_{ij}(M)=\lim_{M\to\infty}\sigma'_{ij}(M)=\sigma'_{ij}.$$

Proof *We continue developing the expression derived in the proof of Theorem A.1 as follows*

$$
\begin{aligned}
& p'_{null|ij}(M) - p'_{null|ik}(M) \\
= \; & \frac{1}{w'q'_{ik}(M)}\left(\alpha_i q'_k(M) + \sum_{h=0}^{C-1}\Delta_{ih}\sigma'_{hk}(M)\right) - \frac{1}{w'q'_{ij}(M)}\left(\alpha_i q'_j(M) + \sum_{h=0}^{C-1}\Delta_{ih}\sigma'_{hj}(M)\right) \\
& + \frac{M}{(M-1)w'q'_{ik}(M)q'_{ij}(M)}\sum_{h=0}^{C-1}\Delta_{ih}q'_h(M)\left(q'_k(M)\sigma'_{ij}(M) - q'_k(M)q'_i(M)I_{\{i=j\}}\right. \\
& \left. -q'_j(M)\sigma'_{ik}(M) + q'_j(M)q'_i(M)I_{\{i=k\}}\right) \\
= \; & \frac{1}{w'q'_{ik}(M)}\alpha_i q'_k(M) + \frac{1}{w'q'_{ik}(M)}w'(1-p'_{null})\left(q_{ik}(M) - q_i(M)q'_k(M)\right) \\
& - \frac{1}{w'q'_{ij}(M)}\alpha_i q'_j(M) - \frac{1}{w'q'_{ij}(M)}w'(1-p'_{null})\left(q_{ij}(M) - q_i(M)q'_j(M)\right) \\
& + \frac{1}{(M-1)w'q'_{ik}(M)q'_{ij}(M)}\left(w'(1-p'_{null})q_i(M) - \alpha_i\right) \\
& \cdot\left(q'_k(M)\sigma'_{ij}(M) - q'_k(M)q'_i(M)I_{\{i=j\}} - q'_j(M)\sigma'_{ik}(M) + q'_j(M)q'_i(M)I_{\{i=k\}}\right) \\
= \; & \frac{1}{w'q'_{ik}(M)}\alpha_i q'_k(M) - \frac{1}{w'q'_{ij}(M)}\alpha_i q'_j(M) \\
& + \frac{1}{w'q'_{ik}(M)}w'(1-p'_{null})\left(q_{ik}(M) - q_i(M)q_k(M) + q_i(M)q_k(M) - q_i(M)q'_k(M)\right) \\
& - \frac{1}{w'q'_{ij}(M)}w'(1-p'_{null})\left(q_{ij}(M) - q_i(M)q_j(M) + q_i(M)q_j(M) - q_i(M)q'_j(M)\right) \\
& + \left(\frac{1}{(M-1)q'_{ik}(M)q'_{ij}(M)}(1-p'_{null})q_i(M) - \frac{1}{(M-1)w'q'_{ik}(M)q'_{ij}(M)}\alpha_i\right) \\
& \cdot\left(q'_k(M)\sigma'_{ij}(M) - q'_k(M)q'_i(M)I_{\{i=j\}} - q'_j(M)\sigma'_{ik}(M) + q'_j(M)q'_i(M)I_{\{i=k\}}\right) \\
= \; & \frac{1}{w'q'_{ik}(M)}\alpha_i q'_k(M) - \frac{1}{w'q'_{ij}(M)}\alpha_i q'_j(M) \\
& + \frac{1}{w'q'_{ik}(M)}w'(1-p'_{null})\left(q_{ik}(M) - q_i(M)q_k(M) + q_i(M)(q_k(M) - q'_k(M))\right) \\
& - \frac{1}{w'q'_{ij}(M)}w'(1-p'_{null})\left(q_{ij}(M) - q_i(M)q_j(M) + q_i(M)(q_j(M) - q'_j(M))\right)
\end{aligned}
$$

$$
\begin{aligned}
&+\left(\frac{1}{(M-1)q'_{ik}(M)q'_{ij}(M)}(1-p'_{null})q_i(M)-\frac{1}{(M-1)w'q'_{ik}(M)q'_{ij}(M)}\alpha_i\right)\\
&\cdot\left(q'_k(M)\sigma'_{ij}(M)-q'_k(M)q'_i(M)I_{\{i=j\}}-q'_j(M)\sigma'_{ik}(M)+q'_j(M)q'_i(M)I_{\{i=k\}}\right)\\
=\;&\frac{1}{w'q'_{ik}(M)}\alpha_i q'_k(M)-\frac{1}{w'q'_{ij}(M)}\alpha_i q'_j(M)\\
&+\frac{1}{q'_{ik}(M)}(1-p'_{null})\left(q_{ik}(M)-q_i(M)q_k(M)\right)+\frac{1}{w'q'_{ik}(M)}q_i(M)\sum_{h=0}^{C-1}\Delta_{CS,h}\sigma'_{hk}(M)\\
&-\frac{1}{q'_{ij}(M)}(1-p'_{null})\left(q_{ij}(M)-q_i(M)q_j(M)\right)-\frac{1}{w'q'_{ij}(M)}q_i(M)\sum_{h=0}^{C-1}\Delta_{CS,h}\sigma'_{hj}(M)\\
&+\left(\frac{1}{(M-1)q'_{ik}(M)q'_{ij}(M)}(1-p'_{null})q_i(M)-\frac{1}{(M-1)w'q'_{ik}(M)q'_{ij}(M)}\alpha_i\right)\\
&\cdot\left(q'_k(M)\sigma'_{ij}(M)-q'_k(M)q'_i(M)I_{\{i=j\}}-q'_j(M)\sigma'_{ik}(M)+q'_j(M)q'_i(M)I_{\{i=k\}}\right).
\end{aligned}
$$

Next, we remove the dependence on k by multiplying by $\frac{q'_{ik}(M)}{q'_i(M)}$ and summing over k as in

$$
\begin{aligned}
&p'_{null|ij}(M)-p'_{null|i}(M)\\
=\;&\sum_{k=0}^{C-1}\frac{q'_{ik}(M)}{q'_i(M)}(p'_{null|ij}(M)-p'_{null|ik}(M))\\
=\;&\sum_{k=0}^{C-1}\frac{1}{w'q'_i(M)}\alpha_i q'_k(M)-\sum_{k=0}^{C-1}\frac{q'_{ik}(M)}{q'_i(M)}\frac{1}{w'q'_{ij}(M)}\alpha_i q'_j(M)\\
&+\sum_{k=0}^{C-1}\frac{1}{q'_i(M)}(1-p'_{null})\left(q_{ik}(M)-q_i(M)q_k(M)\right)+\sum_{k=0}^{C-1}\frac{1}{w'}\frac{q_i(M)}{q'_i(M)}\sum_{h=0}^{C-1}\Delta_{CS,h}\sigma'_{hk}(M)\\
&-\sum_{k=0}^{C-1}\frac{q'_{ik}(M)}{q'_i(M)}\frac{1}{q'_{ij}(M)}(1-p'_{null})\left(q_{ij}(M)-q_i(M)q_j(M)\right)\\
&-\sum_{k=0}^{C-1}\frac{q'_{ik}(M)}{q'_i(M)}\frac{1}{w'q'_{ij}(M)}q_i(M)\sum_{h=0}^{C-1}\Delta_{CS,h}\sigma'_{hj}(M)\\
&+\sum_{k=0}^{C-1}\left(\frac{1}{(M-1)q'_i(M)q'_{ij}(M)}(1-p'_{null})q_i(M)-\frac{1}{(M-1)w'q'_i(M)q'_{ij}(M)}\alpha_i\right)\\
&\cdot\left(q'_k(M)\sigma'_{ij}(M)-q'_k(M)q'_i(M)I_{\{i=j\}}-q'_j(M)\sigma'_{ik}(M)+q'_j(M)q'_i(M)I_{\{i=k\}}\right)
\end{aligned}
$$

$$
\begin{aligned}
= \; & \frac{1}{w' q_i'(M)} \alpha_i - \frac{1}{w' q_{ij}'(M)} \alpha_i q_j'(M) - \frac{1}{w' q_{ij}'(M)} q_i(M) \sum_{h=0}^{C-1} \Delta_{CS,h} \sigma_{hj}'(M) \\
& - \frac{1}{q_{ij}'(M)} (1 - p_{null}') \left(\sigma_{ij}(M) + q_{ij}(M) - q_i(M) I_{\{i=j\}} \right) \frac{1}{M} \\
& + \left(\frac{1}{(M-1) q_i'(M) q_{ij}'(M)} (1 - p_{null}') q_i(M) - \frac{1}{(M-1) w' q_i'(M) q_{ij}'(M)} \alpha_i \right) \\
& \cdot \left(\sigma_{ij}'(M) - q_i'(M) I_{\{i=j\}} + q_j'(M) q_i'(M) \right).
\end{aligned}
$$

Next, we multiply by M and consider the limit as $M \to \infty$ to deduce

$$
\begin{aligned}
& \lim_{M \to \infty} M \left(p_{null|ij}'(M) - p_{null|i}'(M) \right) \\
= \; & \lim_{M \to \infty} \left[\frac{M}{w' q_i'(M)} \alpha_i - \frac{M}{w' q_{ij}'(M)} \alpha_i q_j'(M) - \frac{M}{w' q_{ij}'(M)} q_i(M) \sum_{h=0}^{C-1} \Delta_{CS,h} \sigma_{hj}'(M) \right. \\
& - \frac{1}{q_{ij}'(M)} (1 - p_{null}') \left(\sigma_{ij}(M) + q_{ij}(M) - q_i(M) I_{\{i=j\}} \right) \\
& + \left(\frac{M}{(M-1) q_i'(M) q_{ij}'(M)} (1 - p_{null}') q_i(M) - \frac{M}{(M-1) w' q_i'(M) q_{ij}'(M)} \alpha_i \right) \\
& \left. \cdot \left(\sigma_{ij}'(M) - q_i'(M) I_{\{i=j\}} + q_j'(M) q_i'(M) \right) \right] \\
= \; & \lim_{M \to \infty} \left[\frac{1}{\frac{w'}{M} q_i'(M)} \alpha_i - \frac{1}{\frac{w'}{M} q_{ij}'(M)} \alpha_i q_j'(M) - \frac{1}{\frac{w'}{M} q_{ij}'(M)} q_i(M) \sum_{h=0}^{C-1} \Delta_{CS,h} \sigma_{hj}'(M) \right. \\
& - \frac{1}{q_{ij}'(M)} (1 - p_{null}') \left(\sigma_{ij}(M) + q_{ij}(M) - q_i(M) I_{\{i=j\}} \right) \\
& + \left(\frac{M}{(M-1) q_i'(M) q_{ij}'(M)} (1 - p_{null}') q_i(M) - \frac{1}{(M-1) \frac{w'}{M} q_i'(M) q_{ij}'(M)} \alpha_i \right) \\
& \left. \cdot \left(\sigma_{ij}'(M) - q_i'(M) I_{\{i=j\}} + q_j'(M) q_i'(M) \right) \right] \\
= \; & \frac{1}{\Delta_{MCS} q_i} \alpha_i - \frac{1}{\Delta_{MCS} q_i} \alpha_i - \frac{1}{\Delta_{MCS} q_j} \sum_{h=0}^{C-1} \Delta_{CS,h} \sigma_{hj} \\
& - \frac{1}{q_i q_j} (1 - p_{null}') \left(\sigma_{ij} - q_i I_{\{i=j\}} + q_i q_j \right) + \frac{1}{q_i q_j} (1 - p_{null}') \left(\sigma_{ij}' - q_i I_{\{i=j\}} + q_j q_i \right) \\
= \; & - \frac{1}{\Delta_{MCS} q_j} \sum_{h=0}^{C-1} \Delta_{CS,h} \sigma_{hj} - \frac{1}{q_i q_j} (1 - p_{null}') (\sigma_{ij} - \sigma_{ij}').
\end{aligned}
$$

But notice that $-\frac{1}{\Delta_{MCS}q_j}\sum_{h=0}^{C-1}\Delta_{CS,h}\sigma_{hj}$ *is independent of* i*. So, in fact, if we multiply by* $\frac{q'_{ij}(M)}{q'_j(M)}$ *and sum over* i *we can also show that*

$$\begin{aligned}
&\lim_{M\to\infty} M\left(p'_{null|j} - p'_{null}\right) \\
&= \lim_{M\to\infty}\left[\sum_{i=0}^{C-1}\frac{q'_{ij}(M)}{q'_j(M)}M\left(p'_{null|ij}(M) - p'_{null|i}(M)\right)\right] \\
&= \sum_{i=0}^{C-1}\left(\lim_{M\to\infty}\frac{q'_{ij}(M)}{q'_j(M)}\right)\left(-\frac{1}{\Delta_{MCS}q_j}\sum_{h=0}^{C-1}\Delta_{CS,h}\sigma'_{hj} - \frac{1}{q_iq_j}(1-p'_{null})(\sigma_{ij}-\sigma'_{ij})\right) \\
&= \sum_{i=0}^{C-1} q_i\left(-\frac{1}{\Delta_{MCS}q_j}\sum_{h=0}^{C-1}\Delta_{CS,h}\sigma'_{hj} - \frac{1}{q_iq_j}(1-p'_{null})(\sigma_{ij}-\sigma'_{ij})\right) \\
&= \left(-\frac{1}{\Delta_{MCS}q_j}\sum_{h=0}^{C-1}\Delta_{CS,h}\sigma'_{hj}\right)\sum_{i=0}^{C-1}q_i - \frac{1}{q_j}(1-p'_{null})\sum_{i=0}^{C-1}(\sigma_{ij}-\sigma'_{ij}) \\
&= -\frac{1}{\Delta_{MCS}q_j}\sum_{h=0}^{C-1}\Delta_{CS,h}\sigma_{hj}.
\end{aligned}$$

At this point it is easy to see that the statement

$$\begin{aligned}
\lim_{M\to\infty} M\left(p'_{null|ji} - p'_{null|i}\right) &= -\frac{1}{\Delta_{MCS}q_j}\sum_{h=0}^{C-1}\Delta_{CS,h}\sigma_{hj} - \frac{1}{q_iq_j}(1-p'_{null})(\sigma_{ij}-\sigma'_{ij}) \\
= \lim_{M\to\infty} M\left(p'_{null|j} - p'_{null}\right) &= -\frac{1}{\Delta_{MCS}q_j}\sum_{h=0}^{C-1}\Delta_{CS,h}\sigma_{hj}
\end{aligned}$$

is true if and only if

$$\sigma_{ij} - \sigma'_{ij} = \lim_{M\to\infty}\sigma_{ij} - \lim_{M\to\infty}\sigma'_{ij} = 0.$$

□

Appendix B

Major Software Projects

In order to produce the results for examples given throughout this document, computer programs were developed to perform necessary calculations. In particular, Chapter 5 and Chapter 6 each required substantial pieces of software to be written. For the former, approximately 3000 lines of C++ code was developed to perform analytic calculations of both steady-state and transient measures proposed in the chapter. For the latter, a software simulator consisting of approximately 2500 lines of C++ was designed and used to evaluate our proposed ant algorithm framework. These projects are significant enough that they deserve extra discussion. We will consider each of them separately in this appendix.

B.1 Embedded Urn Analytic Model Calculator

The embedded urn model described in Chapter 5 is a Markov chain. The measures we propose for a given urn require solving for a stationary distribution and determining

the powers of a special single step transition matrix. These calculations involve solving a moderate sized system of linear equations and performing a sequence of matrix multiplications respectively. Both of these are common algebraic operations, and various collections of computer functions have been written to perform these types of calculations. Instead of rewriting these required functions, we have chosen to link to a library called LAPACK (Linear Algebra PACKage). It is widely available free of charge for many common computer platforms. More importantly, the matrix algebra routines contained in LAPACK are mature and thoroughly tested. At the time of writing, version 3.0 of the open source package is available for use. The library itself is written in Fortran, but there is very little trouble interfacing with it from C++, especially if the gcc (GNU compiler collection) is used for all compilation of the code.

The main task of the C++ code is to construct appropriate data structures for the system being studied, pass them in a suitable format for LAPACK work on, and then use the returned results to complete any calculations for the measures outlined in Chapter 5. The most computationally intense part of the process are the calls to LAPACK. Still, any computations we have described can be performed in just a few seconds using a modestly equipped personal computer running our model calculator code. The calculations required for the examples in this document were easily carried out on a 1.6 GHz Intel machine with 512 MB of RAM.

B.2 Ant Routing Framework Simulator

The ant routing framework of Chapter 6 is too sophisticated to be completely modelled analytically. For this reason a software simulator was developed to study the

implementation. It is written in the versatile and efficient C++ programming language. At its heart, the software design is quite simple. Being event driven, the simulator's central data structure is a linked list of pending actions. We ensure that new events are always inserted into the list at the point in which they must occur, as referenced by global simulation time value. In this way, the current action required can easily be read off the head of the linked list. That is, a simulation run proceeds by reading an event from the list of pending events, processing it, which typically involves triggering new events, and then moving on to the next event. Times between events are assumed to be exponentially distributed, and they are randomly generated on the fly using appropriate exponential rate parameters which are input before hand. A given simulation run terminates after a specified amount of simulation time has passed.

The simulator is capable of creating and monitoring a continuous flow of both ants and data packets on a given connected network topology. The network can be arbitrarily specified in terms of the number of nodes and links, but it is assumed that overall the system is homogeneous in the sense that each node runs a copy of the ant routing algorithm with identical parameters. For the purpose of analyzing the ant routing framework, the simulator is structured around a single dramatic traffic load change. Complete pictures of the loading to be used before and after the switch must also be specified upfront. A simulation evolves as described above, with data packet arrival rates for each source and destination pair taken according to the appropriate loading scheme depending on whether the current simulation time falls before or after the specified switch over point. There is an initial phase however, in which ants choose their next-hop completely randomly in order to discover routes. No data traffic is

sent during this phase. Even after data packets begin to flow under the first loading scheme, we do not take any measurements of the network until after a prescribed time. This is necessary since the flow of feedback and the flow of traffic depend on one another. We need to hold off taking any performance samples until the two processes develop a stable relationship.

There are 5 event types possible, each identified by a distinct action, during any simulation. In the following, we give a brief description of each.

1. *Observe output queues.* The dynamic link costs in ant algorithm routing are the delays caused by output queueing as traffic is processed for transmission. We assume that ants can collect a measure of this delay that would be felt by data packets, without actually experiencing it themselves. This means that each node must maintain an estimate of the size of the delays associated with its output queues at all times. At regular intervals signalled by this event, the node updates each estimate by viewing the current unfinished work at each outgoing link. More specifically, the node tracks a number of these delay estimates over a prescribed window of time, and ultimately, the value passed on to ants is the overall average of these.

2. *Ant wave.* In order for the algorithm to work, ants must flow in the network. The ant wave event occurs at regular times to trigger the dispatch of an ant from every node to a randomly chosen destination. The time between waves is configurable. Because of the centralized nature of a software simulator, this method is simple and effective. It is worth noting however that an real ant routing implementation would most likely not work in this fashion as it would require

unnecessary synchronization on the part of nodes. Dispatched ants gather delay information and drop it at their destinations. At a later time, and at least one ant wave in length, the feedback will be transfered back to the source where it is required. This task is also the responsibility of the ants. As they travel they simultaneously disperse feedback information to its intended destination. Once new feedback arrives at a node, which occurs within an instance of an ant wave, updates to the next-hop routing table and route routing table are performed. At the completion of one ant wave, the next one is requested by generating another ant wave event and placing it in the linked list of events.

3. *Data packet arrival.* Various actions must be performed each time a new data packet is admitted to the network at some source node. Chief among these is selecting a route to follow to the desired destination node. The arrival time must also be recorded, and then the data packet is directed to the first-hop on the route by placing it into the appropriate output queue. A data packet step event is issued to handle the data packet from here on. It will not be addressed until the data packet clears the output queue and is completely transmitted to the next node on route. Finally, the end of processing one data packet arrival event involves issuing the next data packet arrival to the node.

4. *Data packet step.* This event is triggered each time a data packet moves from one node to the next along a selected route. The main required action for this event depends on whether or not the data packet has reached its destination node as a result of the last transmission step. If so, measures of delay based on the current time and the original arrival time of the data packet to the network are noted, and the data packet is destroyed. No further events are triggered in

this case. Otherwise, the data packet is directed to its next-hop by placing it in the appropriate output queue. Its progress will be monitored from here on by issuing another data packet step event, which will be processed once the data packet has successfully been transfered to the next node.

5. *Collect performance statistics.* The most important function of the simulator is to evaluate the simulated environment. At regular intervals, performance data is collected for this purpose. Some of the information pertains to a specific set of routes that are determined by selecting a source node, destination node and first-hop node ahead of time. In particular, we focus on only this set when looking at pheromone levels. Other data gathered is global in nature. For instance, we collect network-wide data packet delay values and statistics on the use of ants such as feedback latency and the number of backward ants dispatched. During a simulation the performance data is simply collected into a number of arrays associated with each measure we consider. Any manipulation or processing of the data to produce results occurs just before the simulator exits. Of course at the end of one collect performance statistics event the software requests another event of the same type be processed later.

At the beginning of each simulation, a linked list of initial events is constructed. This starting list includes a single observe output queues event, a single ant wave event and a single collect performance statistics event. It also contains one data packet arrival notice corresponding to each possible source node. These later events are carefully timed so that they only occur after data packet transmission begins in the network. Note that no data packet step actions are needed at the outset of a simulation. They will be generated as required based on data packet arrivals and movements.

Once these initial actions are established, it is easy to see that each event type will be constantly replenished as part of the processing of the type, as described earlier. That is to say, a simulation is self-sustaining once event processing begins. It only stops because processing is suspended at a particular simulation termination time.

Depending on the inter-event times input and the size of the network to be studied, the real time required to run a given simulation can vary widely. The memory requirements are always quite minimal however. As a reference point, the realistic 20 node network considered in Chapter 6 takes approximately 3 minutes to simulate once on a 1.6 GHz Intel machine with 512 MB of RAM. Even without increasing the network size, increasing the traffic loading for instance, can significantly increase the real time required for one simulation run.

A final note on the simulator code is that it automates the process of generating pointwise averaged results from a specified number of individual runs. Averaged results are used extensively in the examples of Chapter 6. To utilize this feature we only need to specify a starting random seed for the pseudo-random number generator, and the desired number of simulations to include in the average. From this information the simulator loops through the necessary number of runs, reinitializing itself each time using a different random seed. Arrays of performance data for each measure are collected over time as before. Now, for a given measure, the corresponding arrays from each simulation run form the rows of a matrix of data. Once all the individual runs are complete, averaged arrays of performance statistics corresponding to each measure are created by averaging each column of the appropriate matrix. These pointwise averaged results are then returned and automatically plotted by the simulator code.

The increase in real time to calculate averaged results is essentially the number of simulation runs multiplied by the time for one run. So for instance, in the case of the 20 node CA*net 4 which takes 3 minutes to simulate on a 1.6 GHz Intel machine with 512 MB of RAM, it takes approximately 5 hours to produce results with 100 simulations incorporated into the average. Technically more RAM is used in this case as well since we store performance information for each simulation run until the simulator completes. Practically however, this slightly increased amount is not noticeable. The limiting factor remains the real time requirements to run the simulator.