Applying Quasi-Separability to Markovian Process Algebra

Nigel Thomas Stephen Gilmore

Department of Computer Science
The University of Edinburgh

{*nat,stg*}*@dcs.ed.ac.uk*

Abstract

Stochastic process algebras have become an accepted part of performance modelling over recent years. Because of the advantages of compositionality and flexibility they are increasingly being used to model larger and more complex systems. Therefore tools which support the evaluation of models expressed using stochastic process algebra must be able to utilise the full range of decomposition and solution techniques available. In this paper we study a class of models which do not give rise to a product form solution but can nevertheless be decomposed into their components without loss of generality. We also exemplify the use of the Markovian process algebra PEPA with the spectral expansion technique which enables a class of PEPA models with infinite state space to be solved numerically.

1 Introduction

The advantages of using a stochastic process algebra to specify performance models are well documented (see Hillston [9] for example). In brief, a process algebra allows models to be compared e.g. for equivalence; to be analysed e.g. to reveal deadlocks or reducible structures; to be constructed at a higher level of abstraction, possibly by designers, rather than performance modellers; and in some cases to be solved automatically, either exactly or through the construction of approximations. The study of stochastic process algebra has now reached a stage whereby we now wish to consider not only how systems are to be specified, but also how complex systems can be simplified and solved automatically. One part of this research has been the study of process algebra models that give rise to product form solutions [6, 10, 17]. However, the class of performance models exhibiting product form is limited. In addition it is not always necessary to derive a full product form decomposition.

Some stochastic process algebra work on product form has been based on widely known cases of product form solution in queueing network models [6, 11]. In this paper we consider an alternative method of model decomposition which can be found in the queueing network literature, *quasi-separability*. This technique has been used in the study of systems which suffer breakdowns. Very little work has been done involving breakdowns affecting more than one queue. Notable exceptions to this are Mitrani et al [16, 20] and Mikou et al [12, 13], none of which give rise to product form solutions. However, in [16] and [20] the models were shown to be *quasi-separable*. Because of that property, one can determine exactly the performance measures in models with many large elements. An individual element can be analysed in isolation from others, provided that a full description of the influence of other elements of the system is included as a state variable.

We illustrate this approach using a model of parallel fault-protected gateways to a network. Messages are generated by users requesting access to the network and are routed through one of the parallel gateways. The gateways are subject to failures that interrupt access to the network for random periods of time. Messages may be directed away from gateways that are known to be faulty, but messages already awaiting access at a faulty gateway are held until repair is complete. No messages are lost. In general the gateways are subject to failures and repairs of different rates. In addition messages are processed at different rates at different gateways, but the service they receive is functionally equivalent. This system is modelled as a system of M/M/1 queues in parallel.

A common misconception is that a process algebra is not a good way to specify queueing problems. One reason for this is that it is very easy to informally specify queueing problems using a common understanding amongst queueing theorists. However, stochastic process algebras provide and "easy to use" interface [15] as well as a formal basis for the analysis and solution of complex stochastic systems. Many examples appear in the literature of queues specified using stochastic process algebras. For the most part these are very simple queueing systems used to illustrate some properties of the process algebra. However, a few papers do exist that use stochastic process algebra to study more complex queueing systems, amongst these are Herzog and Mertsiotakis [7], Bernardo et al [1] and Thomas and Hillston [18, 19].

In the following sections we will give a brief overview of the Markovian process algebra PEPA, introduce the concept of quasi-separability, define the model we wish to study using PEPA and show how it can be solved directly from the PEPA specification using the spectral expansion method.

2 PEPA

PEPA, being a Markovian Process Algebra, only supports actions that occur with rates that are negative exponentially distributed.

Specifications written in PEPA represent Markov processes and can be mapped to a continuous time Markov chain (CTMC) for analytic or numerical solution. In PEPA systems are modelled as an interaction of *components* and *activities*, corresponding to *states* and *transitions* in the underlying Markov process. Each activity has an *action type* (or simply *type*). The duration of each activity is represented by the parameter of the associated exponential distribution: the *activity rate* (or simply *rate*) of the activity. This parameter may be any positive real number, or the distinguished symbol $\top$ (read as *unspecified*).

2.1 Syntax and informal semantics

PEPA provides a small set of combinators. These combinators, together with their names and interpretations, are presented informally below. A much fuller explanation and a specification of the operational semantics of PEPA is given in [9].

Prefix: $(\alpha, r).P$ Prefix is the basic mechanism by which the behaviours of components are constructed. The component carries out activity (α, r) and subsequently behaves as component P.

Constant: $A \stackrel{\mathit{def}}{=} P$ Constants are components whose meaning is given by a defining equation; $A \stackrel{\mathit{def}}{=} P$ gives the constant A the behaviour of the component P. This is how we assign names to components (behaviours). There is no explicit recursion operator but components of infinite behaviour may be readily described using sets of mutually recursive defining equations.

Choice: $P+Q$ The component represents a system which may behave either as component P or as Q: all the current activities of both components are enabled. The first activity to complete, determined by a race condition, distinguishes one component, the other is discarded. The choice combinator represents competition between components.

Hiding: $P \backslash L$ The component behaves as P except that any activities of types within the set L are hidden, i.e. such an activity exhibits the unknown type $\top$ and the activity can be regarded as an internal delay by the component.

Cooperation: $P \underset{L}{\bowtie} Q$ The components proceed independently with any activities whose types do not occur in the *cooperation set* L. The activities not in L are *individual activities*. However, activities with action types in the set L require the simultaneous involvement of both components (*shared activities*). These activities are only enabled in $P \underset{L}{\bowtie} Q$ when they are enabled in both P and Q.

The published MPAs differ on how the rate of shared activities are defined [8]. In PEPA the shared activity occurs at the rate of the slowest participant. If an activity has an unspecified rate in a component, the component is passive with respect to that action type. This means that the component does not influence the rate at which any shared activity occurs. The cooperation combinator associates to the left but brackets may also be used to clarify the meaning. The parallel combinator $\|$ is used as shorthand to denote synchronisation with no shared activities, i.e. $P \| Q \equiv P \underset{\emptyset}{\bowtie} Q$.

2.2 Execution strategy

A race condition governs the dynamic behaviour of a model whenever more than one activity is enabled. This has the effect of replacing the non-deterministic branching of classical process algebra with probabilistic branching. The probability that a particular activity completes is given by the ratio of the activity rate to the sum of the activity rates of all the enabled activities. Any other activities which were simultaneously enabled will be interrupted or aborted. The memoryless property of the exponential distribution makes it unnecessary to record the remaining lifetime in either case.

3 Quasi-Separability

Consider a system which consists of N separate components where the state of each component i can be described by the pair (X_i, Y_i). The state of the whole system therefore can be described by the pair of vectors $(\mathbf{X}, \mathbf{Y})$, where $\mathbf{X} = (X_1, X_2, \ldots, X_N)$ and $\mathbf{Y} = (Y_1, Y_2, \ldots, Y_N)$. If it is possible to analyse the behaviour of each component, i, of the system exactly by only considering those variables that describe it, i.e. (X_i, Y_i), then the system is said to be *separable*. In this case all the components are independent and a product form solution exists. For the system to be *quasi-separable* it is necessary only that it is possible to analyse the behaviour of each component, i, of the system exactly by only considering one of the pair of system state vectors and the single variable from the other vector that is related to component i, i.e. either $(\mathbf{X}, Y_i)$ or $(X_i, \mathbf{Y})$. A model of N such components, $(\mathbf{X}, \mathbf{Y})$, may be reduced in this way to N submodels, $(\mathbf{X}, Y_i)$, if the rates of actions which change the state of Y_i are determined only by the current states of Y_i and $\mathbf{X}$ and the rates of actions which change the state of X_i are determined only by the current state of $\mathbf{X}$.

The analysis of these submodels gives rise to expressions for their steady-state marginal probabilities. As stated above, these marginal probabilities do not, in general, give rise to expressions for the joint probability of the whole system, i.e. no product form solution exists. However, it is still possible to calculate many interesting performance measures, such as the average response time of the system, using these marginal probabilities.

Consider such a system of N components expressed in PEPA. The components are modelled as PEPA components, $A_1, \ldots, A_N$ say. The interactions of these components are co-ordinated by another component B, which we will refer to as the scheduler. These components are related to the state space of the underlying CTMC of the form discussed above such that, B represents all the states $\mathbf{X}$ and A_i represents at least all the states Y_i, $i = 1, \ldots, N$. That is, the A_i's represent those parts of the system that are considered in isolation and B represents those parts which are not. Note that in general each component A_i may represent more than just Y_i as it may be desirable to include information that restricts the set of possible legal pairs $(\mathbf{X}, Y_i)$.

No actions are synchronised between the components $A_1, \ldots, A_N$, but the state of A_i is determined by actions that are synchronised between it and B, contained in the set L. The state of the scheduler, B, is determined only by actions which can be internalised. That is, actions which are individual actions of B or which are shared actions such that B determines the rate of the action and the action cannot be blocked by the state of A_i. Thus the system can be described in the following way;

$$(A_1 \parallel \cdots \parallel A_N) \underset{L}{\bowtie} B$$

The set of synchronised actions L can be rewritten as N subsets L_i, $i = 1, \ldots, N$. The subset L_i contains only all the actions that are synchronised between B and A_i. Denote by C_i the set of all actions that are synchronised between B and each A_j such that $j \neq i$, i.e.

$$C_i = \bigcup_{\forall j \neq i} L_j$$

Clearly, $C_i \subset L$ and $L = (L_i \cup C_i)$, $i = 1, \ldots, N$. If the actions in the subsets C_i are such that their rates are not influenced by A_i and the actions in the intersection $L_i \cap C_i$ are such that their rates are not influenced by A_k, $k = 1, \ldots, N$, then the system can be rewritten in the following way;

$$(A_1 \underset{L_1}{\bowtie} B) \underset{C_1}{\bowtie} (A_2 \parallel \cdots \parallel A_N)$$

This is done without changing the underlying CTMC except in the order in which state space is explored i.e. the rewritten model is *isomorphic* to the original model. The proof of this equivalence follows from the equational laws for isomorphic components and expansion law given in [9] and the restrictions on the sets L, L_1 and C_1 given above. In addition the subsystem,

$$(A_1 \underset{L_1}{\bowtie} B)$$

can now be isolated from the system without altering the individual behaviour of A_1 in any way on the conditions for L_1 and C_1 given above. The same approach can be applied for every A_i, L_i and C_i to give expressions for all N subsystems. A system that can be treated in such a way satisfies the conditions for *quasi-separability* presented above. Clearly the characterisation presented here is in an idealised form and requires further development for much more general models to be considered.

3.1 The PEPA Analyser

The purpose of confining the expression of a performance model to a particular modelling notation is so that the model can be automatically checked for the presence or absence of certain properties, among them quasi-separability. We are presently completing the implementation of a separability analyser for PEPA.

The PEPA Analyser uses some of the routines which were developed for the PEPA Workbench together with static analysis routines taken from the PEPA-to-Ada translator. Descriptions of those tools are available elsewhere [3, 4]. The static analysis routines are used in the inference of the interfaces which model components present to the rest of the system and also in determining the smallest cooperation sets for each pair of components in the model, considered in isolation.

Given this information the PEPA Analyser classifies components as being either *slaves*—because their interaction is controlled by another component—or *schedulers*. Upon being able to build up a set of non-communicating slaves controlled by a scheduler the PEPA Analyser identifies a re-configuration of the system equation which isolates a quasi-separable component. If the Analyser is unable to form a set of slaves controlled by a scheduler it announces that the system is not quasi-separable.

4 Example

In this section we illustrate the application of quasi-separability using an example from the field of networking. The model presented here is shown to be of the type defined in Section 3. The model is decomposed into a set of submodels which are solved numerically using the spectral expansion technique. Finally some numerical results are presented.

4.1 Model Description

We consider a system where jobs from a common incoming stream may be directed to one of N alternative gateways, each of which consists of a single server and an unbounded queue. The service, breakdown and repair processes at the different nodes are independent of each other and have different parameters, in general. The consequences of a breakdown at a server are not too catastrophic: service stops and the existing jobs remain in place; new arrivals during the subsequent repair period may be re-directed to other queues. There are no job losses.

Jobs arrive into the system in a Poisson stream with rate λ. There are N servers, each with an associated unbounded queue, to which incoming jobs may be directed. Server k goes through alternating independent operative and inoperative periods, distributed exponentially with means $1/\xi_k$ and $1/\eta_k$, respectively. While it is operative, the jobs in its queue receive exponentially distributed services with mean $1/\mu_k$, and depart upon completion. When a server becomes inoperative (breaks down), the corresponding queue, including the job in service (if any), remains in place. Services that are interrupted in this way are eventually resumed from the point of interruption.

The *system configuration* at any moment is specified by the subset, σ, of servers that are currently operative (that subset may be empty, or it may be the set of all servers): $\sigma \subset \Omega_N$, where $\Omega_N = \{1, 2, \ldots, N\}$. There are of course 2^N possible system configurations. If, at the time of arrival, a new job finds the system in configuration σ, then it is directed to node k with probability $q_k(\sigma)$. These decisions are independent of each other, of past history and of the sizes of the various queues. Thus, a routing policy is defined by specifying 2^N vectors,

$$\mathbf{q}(\sigma) = [q_1(\sigma), q_2(\sigma), \ldots, q_N(\sigma)] \ , \ \sigma \subset \Omega_N \ , \tag{4.1}$$

where $q_k(\sigma)$ is the probability that a job is directed to the kth queue, such that for every σ,

$$\sum_{k=1}^{N} q_k(\sigma) = 1$$

Intuitively, it seems better not to send jobs to nodes where the server is inoperative, unless that is unavoidable. This suggests the following strategy. If the subset of operative servers in the current system configuration is σ, and that subset is non-empty, send jobs to node k only if $k \in \sigma$, with probability proportional to q_k :

$$q_k(\sigma) = \frac{q_k}{\sum_{\ell \in \sigma} q_\ell} \ , \ k \in \sigma \ ,$$

$q_k(\sigma) = 0$, otherwise. If σ is empty (i.e. all servers are broken), send jobs to node k with probability q_k $(k = 1, 2, \ldots, N)$. This last point ensures that no jobs are lost and preserves the system property of being fault protected, i.e. the effect of a failure is not catastrophic.

We now wish to specify this problem in the Markovian process algebra PEPA. There are several possible reasons for wanting to do this, some of which were outlined in the introduction. Ultimately this approach will enable us to use an extended version of the PEPA workbench [3] to analyse the structure of the model and obtain a numerical solution direct from a formal specification. Such a specification is given in Figure 1, for the case where there are 2 queues in parallel.

$$
\begin{aligned}
Queue1_0 &\stackrel{def}{=} (arr1, \top).Queue1_1 \\
Queue1_j &\stackrel{def}{=} (arr1, \top).Queue1_{j+1} + (serv1, \top).Queue1_{j-1} \quad , \quad 1 \leq j \leq \infty \\
Queue2_0 &\stackrel{def}{=} (arr2, \top).Queue2_1 \\
Queue2_j &\stackrel{def}{=} (arr2, \top).Queue2_{j+1} + (serv2, \top).Queue2_{j-1} \quad , \quad 1 \leq j \leq \infty \\
Sigma_0 &\stackrel{def}{=} (arr1, \lambda q_1).Sigma_0 + (arr2, \lambda q_2).Sigma_0 + (repair1, \eta_1).Sigma_1 \\
&\quad + (repair2, \eta_2).Sigma_2 \\
Sigma_1 &\stackrel{def}{=} (serv1, \mu_1).Sigma_1 + (arr1, \lambda).Sigma_1 + (fail1, \xi_1).Sigma_0 \\
&\quad + (repair2, \eta_2).Sigma_3 \\
Sigma_2 &\stackrel{def}{=} (serv2, \mu_2).Sigma_2 + (arr2, \lambda).Sigma_2 + (fail2, \xi_2).Sigma_0 \\
&\quad + (repair1, \eta_1).Sigma_3 \\
Sigma_3 &\stackrel{def}{=} (arr1, \lambda q_1).Sigma_3 + (arr2, \lambda q_2).Sigma_3 + (serv1, \mu_1).Sigma_3 \\
&\quad + (serv2, \mu_2).Sigma_3 + (fail1, \xi_1).Sigma_2 + (fail2, \xi_2).Sigma_1
\end{aligned}
$$

$$
(Queue1_0 \parallel Queue2_0) \underset{\{arr1, arr2, serv1, serv2\}}{\bowtie} Sigma_3
$$

Figure 1: A PEPA model of two $M/M/1$ queues in parallel with state dependent routing

As with most languages, there are many ways to describe this model in PEPA. However, we wish not only to produce a concise specification, but also one that captures our view of this model and facilitates simplification and solution in a clear manner. We therefore propose a specification consisting of passive buffers controlled by a scheduler which represents the operational state of the system, σ. The index of the scheduler is an integer representation of the binary system operational state, σ, where broken is 0 and not broken is 1, e.g. if only server i is working the operational (scheduler) state is 2^i and so on. The scheduler states can readily be generated recursively, as can expressions for additional queues. Therefore the description of the system given here can easily be expanded. An alternative approach to specifying such a model might be to consider each of the servers as separate components, rather than collectively in the scheduler. Such an approach would conceivably give rise to fewer terms in the specification when the number of queues is large, but would inevitably make any decomposition based on quasi-separability somewhat harder.

4.2 Deriving Submodels

The model presented here is multi-dimensional, its birth-death processes operate on more than one numerical line. In all cases the state of the queues is dependent on the actions of the scheduler, therefore they are not separable. However, the queues themselves can be considered independent of each other and still maintain the same overall behaviour, i.e. this model exhibits quasi-separability. By considering the queues this way it is possible to find the marginal queue size probabilities, which can then be used to find most performance measures of interest.

The actions $arr1$ and $serv1$ apply only to queue 1 and the actions $arr2$ and $serv2$ apply only to queue 2. Hence the expressions,

$$(Queue1_0 \parallel Queue2_0) \underset{(arr1,arr2,serv1,serv2)}{\bowtie} Sigma_3$$

$$(Queue1_0 \underset{(arr1,serv1)}{\bowtie} Sigma_3) \underset{(arr2,serv2)}{\bowtie} Queue2_0$$

and

$$(Queue2_0 \underset{(arr2,serv2)}{\bowtie} Sigma_3) \underset{(arr1,serv1)}{\bowtie} Queue1_0$$

are isomorphic. Thus, evaluating the first part of each of the last two expressions,

$$Queue(k)_0 \underset{(arr(k),serv(k))}{\bowtie} Sigma_3$$

will give an exact result for the marginal queue size probabilities for queue k. Clearly if models have been specified with passive actions in the scheduler then the scheduler expressions will need to be rewritten before evaluating the synchronised process. The condition by which this method may be applied here is that the state of the scheduler cannot be determined by the number of jobs in the queues and that the number of jobs in each queue cannot be determined by the number of jobs in any other queue.

It is important to note that this approach does not give rise to expressions for the joint queue size probabilities since, as stated, this system is not separable. However the average number of jobs in the system is given by the sum of the average number of jobs in each queue which can be found from the marginal queue size probabilities. Other performance measures of interest can also be derived from these marginal probabilities. It is clear therefore that this approach does not give a product form solution, but is very useful nevertheless.

One further problem remains if we are to obtain a numerical solution to this model, namely that the state space of this model is infinite and the PEPA Workbench can only solve finite state systems. Hence we introduce the use of the spectral expansion solution method.

4.3 Solution by Spectral Expansion

There are several methods by which a model such as the one described here can be solved. One such method is spectral expansion [2] and some evidence has been suggested in [14] which favours its use in this case. In addition spectral expansion has previously been used with the stochastic process algebra TIPP [15]. The results derived for TIPP and spectral expansion are readily applied to PEPA and the model presented here is consistent with restrictions imposed in [15]. Therefore we do not intend to reproduce these earlier results, but rather show how our model is solved using spectral expansion directly from its specification in PEPA.

The spectral expansion method has been widely used to solve models with a 2 dimensional state space which is finite in one dimension and infinite in the other. Models can be solved in this way if, above a certain finite threshold, the rate of change of state is independent of the current state. The reduced form of the model derived in the previous section is of this kind.

In the model considered here the dimensions of the process are clearly evident, $Queue(k)$ is infinite and $Sigma$ is finite. Furthermore every scheduler state $Sigma_i$ forms a legal pair with every state of $Queue(k)_j$ and the service and arrival rates are independent of the number of jobs in the queue (except there is no service when the queue is empty). It is then an easy matter to form 3 matrices, A, B and C, from the PEPA specification corresponding to changes in scheduler state $Sigma_i$, increases in queue size and decreases in queue size:

- Let $A = a_{i,i\prime}$ $(i, i\prime = 0, 1, \ldots, 2^N - 1)$ be the matrix of instantaneous transition rates corresponding to transitions between states of the scheduler, $Sigma_i$ to $Sigma_{i\prime}$. In the above model these transitions correspond to failures and repairs of individual servers in the system.
- Let B be the diagonal matrix whose ith element is equal to the arrival rate when the scheduler is in state $Sigma_i$, i.e. $\lambda q_k(\sigma)$.
- Let C be the diagonal matrix whose ith element is equal to the service rate when the scheduler is in state $Sigma_i$. In this model the service rate is either μ_k or zero, depending on whether the server is operative or not.

In addition, let D be the diagonal matrix whose ith element is equal to the ith row sum of A, the total rate at which servers fail or are repaired when the scheduler is in state $Sigma_i$. Define the (row) vector of equilibrium probabilities of all states with j jobs in the queue:

$$\mathbf{v}(j) = [p(0, j), p(1, j), \ldots, p(2^N - 1, j)] \ , \ \ j = 0, 1, \ldots \qquad (4.2)$$

When $j > 0$, these vectors (4.2) satisfy the balance equations

$$\mathbf{v}(j)(D+B+C) = \mathbf{v}(j)A + \mathbf{v}(j-1)B + \mathbf{v}(j+1)C \quad , \quad j = 1, 2, \ldots \tag{4.3}$$

For $j = 0$, the equation is slightly different:

$$\mathbf{v}(0)(D+B) = \mathbf{v}(0)A + \mathbf{v}(1)C \tag{4.4}$$

In addition, all probabilities must sum up to 1:

$$\sum_{j=0}^{\infty} \mathbf{v}(j)\mathbf{e} = 1 \; , \tag{4.5}$$

where $\mathbf{e}$ is a column vector with 2^N elements, all of which are equal to 1.

The balance equations (4.3) can be rewritten in the form

$$\mathbf{v}(j)Q_0 + \mathbf{v}(j+1)Q_1 + \mathbf{v}(j+2)Q_2 = \mathbf{0} \quad , \quad j = 0, 1, \ldots \tag{4.6}$$

where $Q_0 = B$, $Q_1 = A - D - B - C$ and $Q_2 = C$. This is a homogeneous vector difference equation of order 2, with constant coefficients. Associated with it is the characteristic matrix polynomial, $Q(z)$, defined as $Q(z) = Q_0 + Q_1 z + Q_2 z^2$.

Denote by z_ℓ and $\boldsymbol{\psi}_{\boldsymbol{\ell}}$ the *generalised eigenvalues and left eigenvectors* of $Q(z)$. These quantities satisfy $\boldsymbol{\psi}_{\boldsymbol{\ell}} Q(z_\ell) = \mathbf{0}$, $\ell = 1, 2, \ldots, d$, where $d = degree\{det[Q(z)]\}$. The eigenvalues do not have to be simple, but it is assumed that if z_ℓ has multiplicity r, then it has r linearly independent left eigenvectors. This is invariably observed to be the case in practice. Under that assumption, any solution of (4.6) is of the form

$$\mathbf{v}(j) = \sum_{\ell=1}^{d} x_\ell \boldsymbol{\psi}_{\boldsymbol{\ell}} z_\ell^j \quad , \quad j = 0, 1, \ldots$$

where x_ℓ ($\ell = 1, 2, \ldots, d$), are arbitrary (complex) constants.

Moreover, since only normaliseable solutions are acceptable, if $|z_\ell| \geq 1$ for some ℓ, then the corresponding coefficient x_ℓ must be set to 0. Numbering the eigenvalues of $Q(z)$ in increasing order of modulus, the spectral expansion solution of equation (4.6) can be written as

$$\mathbf{v}(j) = \sum_{\ell=1}^{c} x_\ell \boldsymbol{\psi}_{\boldsymbol{\ell}} z_\ell^j \quad , \quad j = 0, 1, \ldots \tag{4.7}$$

where c is the number of eigenvalues strictly inside the unit disk (each counted according to its multiplicity). In the numerical experiments carried out with this model, the eigenvalues and eigenvectors of $Q(z)$ have always been observed to be simple, real and positive.

Substituting (4.7), for $j = 0$ and $j = 1$, into (4.4), yields a set of homogeneous linear equations for the unknown coefficients x_ℓ. There are $2^N - 1$ independent equations in this set (rather than 2^N) because the generator matrix of the Markov process is singular. A further, non-homogeneous equation is provided by (4.5), which now becomes

$$\sum_{\ell=1}^{2^N} \frac{x_\ell \boldsymbol{\psi}_{\boldsymbol{\ell}} \mathbf{e}}{1 - z_\ell} = 1$$

These equations can be solved uniquely for the coefficients x_ℓ, if $c = 2^N$. This turns out to be the case when the total arrival rate is less than the total service rate over all states of the scheduler. Indeed, this ergodicity condition is equivalent to the requirement that $Q(z)$ has exactly 2^N eigenvalues strictly inside the unit disk.

Given the probabilities $p(i,j)$, the average size of the queue is obtained from

$$L = \sum_{j=1}^{\infty} j \sum_{i=0}^{2^N - 1} p(i,j) \tag{4.8}$$

Hence, having determined the coefficients x_ℓ, the average number of jobs in queue is obtained by substituting (4.7) into (4.8):

$$L = \sum_{\ell=1}^{2^N} \frac{x_\ell z_\ell \boldsymbol{\psi}_{\boldsymbol{\ell}} \mathbf{e}}{(1 - z_\ell)^2} \tag{4.9}$$

Clearly this spectral expansion solution method will need to be carried out for every sub-model obtained using quasi-separability to find the average number of jobs in the system as a whole. It is then a simple matter to find the average response time of the system using Little's Theorem.

4.4 Numerical Results

The performance of any given configuration of gateways is greatly dependent on the chosen routing vector **q** and the external arrival rate, λ. It is therefore a matter of considerable interest to optimise the routing vector with respect to given performance measures. Such a scenario for a 2 queue system is illustrated in Figure 2. Here the average response time of the system is given as a function of the routing vector $(q, 1-q)$ for a number of different values of λ given the same server characteristics. It is clear that a poor choice of routing vector, i.e. q is small, results in a relatively poor level of service for the users. In addition when the load is light the optimum routing vector is likely to favour the faster server, whereas when the load is increased the load is balanced between the servers.

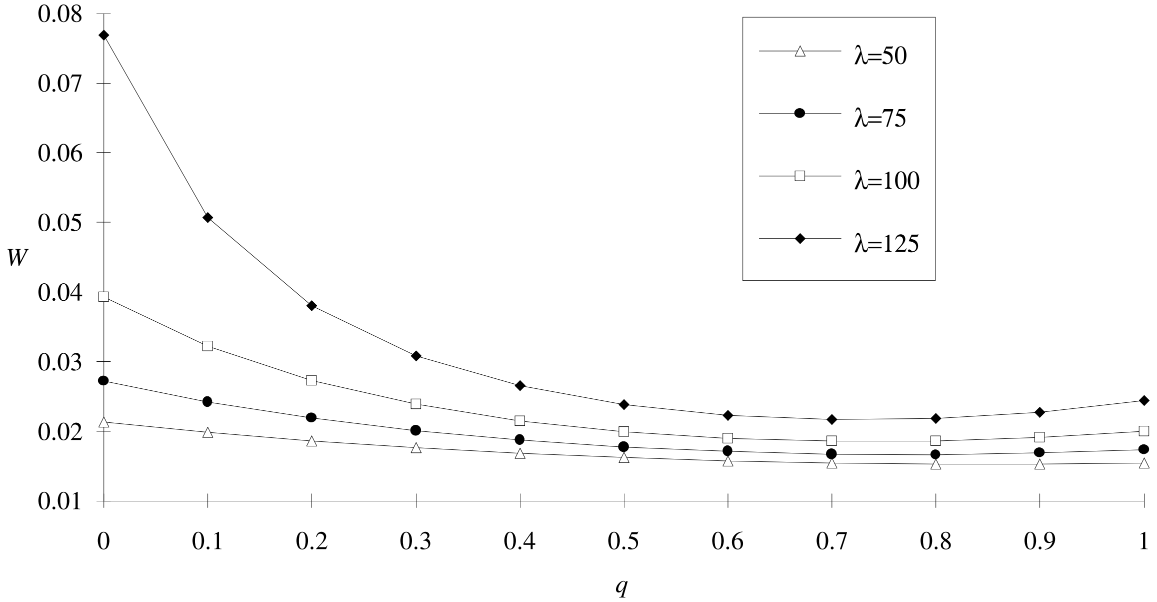

Figure 2: Average response time, W, as a function of routing vector $(q, 1-q)$
$\mu_1 = 250$, $\mu_2 = 150$, $\xi_1 = \xi_2 = 0.1$, $\eta_1 = \eta_2 = 1$

The values of failure and repair rates, ξ and η also greatly affect the performance of the system. In general, frequent but short periods of inoperation cause much less disruption than infrequent but long ones. This is illustrated in Figure 3, where the availability of each server does not change, but the duration of failure and repair periods does. In this example there is little difference in performance between the two cases when the repair periods are shortest. However, when the repair periods are extended the average response time of the system is greatly affected and the consequence of not optimising the routing vector is much more pronounced.

5 Concluding Remarks

In this paper we have shown how complex models expressed in PEPA can be greatly simplified by applying quasi-separability. Decompositions of this kind are extremely useful when tackling models with large state spaces, especially when the state space grows exponentially with the addition of additional components. In the model used here to demonstrate this process the state space is infinite in N dimensions and therefore not generally tractable. However, the decomposition gives rise to N sub-models each of which was infinite in one dimension only.

Quasi-separability can be applied to a wide range of models to derive numerical results very efficiently. While it does not generally give rise to expressions for joint probability distributions it does provide exact results for many performance measures, possibly negating the need for more complex numerical analysis. Also the PEPA Analyser provides support so that models that exhibit quasi-separability can be identified automatically. As such it is a very useful means of reducing the state space of large models.

The characterisation we have derived here, both in general terms and in the context of PEPA is extremely limited. Clearly we need to consider more general forms of this property and this work is

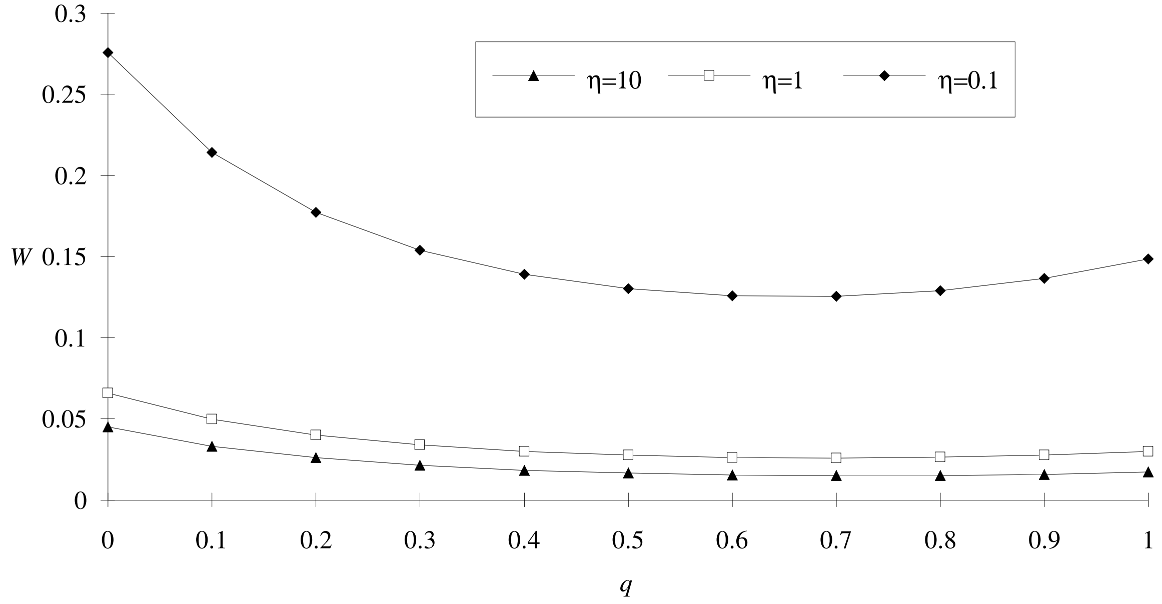

Figure 3: Average response time, W, as a function of routing vector $(q, 1-q)$
$\mu_1 = 150$, $\mu_2 = 100$, $\lambda = 75$, $\eta_1 = \eta 2 = \eta$, $\xi_1 = \xi_2 = \eta/10$

currently ongoing. In addition the example presented here is similarly lacking in generality, being merely a PEPA model of a queueing system. This work forms part of an ongoing effort within our group to characterise and exploit product form and near-product form results in PEPA.

Acknowledgements

The authors wish to thank Jane Hillston for helpful comments during the continued development of this work. Nigel Thomas would like to thank Prof. Isi Mitrani for introducing him to this area and for the initial insight which led to this approach. Nigel Thomas is supported by the Engineering and Physical Sciences Research Council via the COMPA project (G/L10215). Stephen Gilmore is supported by the 'Distributed Commit Protocols' grant from the EPSRC and by Esprit working group FIREworks.

References

[1] M. Bernardo, L. Donatiello and R. Gorrieri, Describing Queueing Systems with MPA, Technical Report UBLCS-94-11, University of Bologna, May, 1994.

[2] R. Chakka and I. Mitrani, A Numerical Solution Method for Multiprocessor Systems with General Breakdowns and Repairs, in: R. Pooley and J. Hillston eds., *Computer Performance Evaluation '92: Modelling Techniques and Tools*, Edinburgh University Press, 1992.

[3] S. Gilmore and J. Hillston, The PEPA Workbench: A Tool to Support a Process Algebra-based Approach to Performance Modelling, in: G. Haring and G. Kotsis eds., *Computer Performance Evaluation: Modelling Techniques and Tools*, Lecture Notes in Computer Science, vol. 794, pp. 353-368, Springer Verlag, 1994.

[4] S. Gilmore, J. Hillston and R. Holton, From SPA models to programs, in: M. Ribaudo ed., Proceedings of the 4th Workshop on Process Algebra and Performance Modelling, Università di Torino, July, 1996.

[5] N. Götz, U. Herzog and M. Rettelbach, TIPP - a language for timed processes and performance evaluation, in: J. Hillston and F. Moller eds., *Proceedings of the Workshop on Process Algebra and Performance Modelling*, University of Edinburgh, UK, May, 1993.

[6] P. Harrison and J. Hillston, Exploiting quasi-reversible structures to find product form solutions in Markovian process algebra models, *The Computer Journal*, **38**(7), pp. 510-520, 1995.

[7] U. Herzog and V. Mertsiotakis, Stochastic Process Algebras Applied to Failure Modelling, in U. Herzog and M. Rettelbach eds., *Proceedings of the 2nd Workshop on Process Algebra and Performance Modelling*, Regensberg, July, 1994.

[8] J. Hillston, The Nature of Synchronisation, in: U. Herzog and M. Rettelbach eds., *Proceedings of the 2nd Workshop on Process Algebra and Performance Modelling*, Regensburg, July, 1994.

[9] J. Hillston, *A Compositional Approach to Performance Modelling*, Cambridge University Press, 1996.

[10] J. Hillston and N. Thomas, Product Form Solution for a class of PEPA Models, in *Proceedings of IEEE International Computer Performance and Dependability Symposium*, Durham NC, September, 1998.

[11] J. Hillston and N. Thomas, A Syntactic Analysis of Reversible PEPA Processes, in *Proceedings of 6th International Workshop on Process Algebra and Performance Modelling*, Nice, September, 1998.

[12] N. Mikou, A Two-Node Jackson Network Subject to Breakdowns, *Stochastic Models*, **4**, pp. 523-552, 1988.

[13] N. Mikou, O. Idrissi-Kacemi and S. Saadi, Two Processors Only Interacting During Breakdown: The Case Where the Load is Not Lost, *Queueing Systems*, **19**, pp. 301-317, 1995.

[14] I. Mitrani and R. Chakka, Spectral Expansion Solution for a Class of Markov Models: Application and Comparison with the Matrix-Geometric Method, *Performance Evaluation*, **23**, pp. 241-260, 1995.

[15] I. Mitrani, A. Ost and M. Rettelbach, TIPP and the Spectral Expansion Method, in: F. Baccelli, A. Jean-Marie and I. Mitrani, eds., *Quantitative Methods in Parallel Systems*, pp. 99-113, Springer-Verlag, 1995.

[16] I. Mitrani and P.E. Wright, Routing in the Presence of Breakdowns, *Performance Evaluation*, **20**, pp. 151-164, 1994.

[17] M. Sereno, Towards a product form solution for stochastic process algebras, *The Computer Journal*, **38**(7), pp. 622-632, 1995.

[18] N. Thomas and J. Hillston, Using Markovian Process Algebra to Specify Interactions in Queueing Systems, Technical Report ECS-LFCS-97-373, University of Edinburgh, 1997.

[19] N. Thomas and J. Hillston, Markovian Queueing Systems modelled with PEPA, in: R.J. Pooley and N. Thomas, eds., *Proceedings of 14th UK Performance Engineering Workshop*, Edinburgh, July, 1998.

[20] N. Thomas and I. Mitrani, Routing Among Different Nodes Where Servers Break Down Without Losing Jobs, in: F. Baccelli, A. Jean-Marie and I. Mitrani, eds., *Quantitative Methods in Parallel Systems*, pp. 248-261, Springer-Verlag, 1995.