An approximate solution of PEPA models using component substitution

Nigel Thomas
nigel.thomas@durham.ac.uk
Research Institute in Software Evolution
University of Durham, UK

Jeremy Bradley and David Thornley
{jb,djt}@doc.ic.ac.uk
Department of Computing
Imperial College London, UK

Abstract

Performance models specified using compositional algebra suffer the well-known state space explosion problem, where a relatively small definition leads to a Markov chain with a large state space that is problematic to solve. As a result it is widely recognised that the development of techniques to solve performance models efficiently is of particular practical importance. Recently the notion of behavioural independence was introduced to exploit the structure of Markovian process algebra models in order to solve models in a compositional manner. In this paper the opposite property, namely control, is used to solve models by substituting components in the model with simpler versions.

1 Introduction

Stochastic process algebra are widely recognised as a good way of specifying performance models since they allow the modeller to exploit the compositionality of systems in the specification. However, this advantageous property for specification often leads to difficulty in solution, as a model with several components will generally lead to a large underlying continuous time Markov chain (CTMC) that requires a significant amount of effort to solve. As a result a key research topic in recent years has been to identify efficient methods for analysing and solving stochastic process algebra models by decomposition (see [5, 6]). This means using the component structure of models to drive the solution, ideally generating isolated solutions for individual components that can be combined to solve the entire model without needing to generate the global state space of the complete model. One element of this work has been the identification and characterisation of product form solutions, using properties such as reversibility [7] and quasi-reversibility [3]. Another focus has been the study of classes of model, which do not have a product form solution, but can nevertheless be decomposed under certain conditions to provide solutions to certain global measures.

Recently a property referred to as *behavioural independence* has been exploited by Thomas [8, 9] to give clear definitions for certain restricted classes of model subject to decompositional solution. Essentially this property states that the behaviour of a particular group of components in a model is not influenced by the behaviour of other components in the model. This non-trivial property can be a powerful tool in identifying independent and semi-independent behaviours. If behavioural independence does not exist then by definition another component must be exerting *control*. In this paper we exploit the existence of control in a model by replacing a component in the model with a simpler version of itself. This simple component offers the same set of interactions as the component it replaced, but in general has far fewer states. Thus the resultant model gives rise to a CTMC that is also much reduced in size and consequently easier to solve. However, in general it is not always possible to know the rates of all transitions within the simple component. Therefore an iterative approach is used to find a convergence between two or more reduced models. This approach is inspired by Gribaudi and Sereno [2] who used a similar technique when solving certain Petri net models.

In the following sections PEPA is summarised and a number of definitions are made, including that of control. The class of model to be studied is then described and the simple component and the solution method specified. To illustrate the approach a simple queueing example is presented followed by some concluding remarks.

2 PEPA

A formal presentation of PEPA is given in [4], in this section a brief informal summary is presented. PEPA, being a Markovian Process Algebra, only supports actions that occur with durations that are negative exponentially distributed. Specifications written in PEPA represent Markov processes and can be mapped to a continuous time Markov chain (CTMC). Systems are specified in PEPA in terms of *activities* and *components*. An activity (α, r) is described by the *type* of the activity, α, and the *rate* of the associated negative exponential distribution, r. This rate may be any positive real number, or given as *unspecified* using the symbol $\top$. The syntax for describing components is given as:

$$P ::= (\alpha, r).P \mid P + Q \mid P/L \mid P \underset{L}{\bowtie} Q \mid A$$

The component $(\alpha, r).P$ performs the activity of type α at rate r and then behaves like P. The component $P + Q$ behaves either like P or like Q, the resultant behaviour being given by the first activity to complete. The component P/L behaves exactly like P except that the activities in the set L are concealed, their type is not visible and instead appears as the unknown type τ. Concurrent components can be synchronised, $P \underset{L}{\bowtie} Q$, such that activities in the *cooperation set* L involve the participation of both components. In PEPA the shared activity occurs at the slowest of the rates of the participants and if a rate is unspecified in a component, the component is passive with respect to that activities of that type. The parallel combinator $\|$ is used as shorthand to denote synchronisation with no shared activities, i.e. $P \| Q \equiv P \underset{\emptyset}{\bowtie} Q$. $A \stackrel{def}{=} P$ gives the constant A the behaviour of the component P.

2.1 Definitions

In the following sections a number of properties of PEPA models will be referred to. Since the definitions of these properties are presented in detail elsewhere [4], only a general description of them is included here. Informally a *derivative* is the "state" of a component defining the current behaviour of a model or component. For example, if $P \stackrel{def}{=} (\alpha, r).Q$ then Q is a derivative of P and if $Q \stackrel{def}{=} (\beta, r_2).R$ then R is a derivative of both Q and P, and so on. The *derivative set*, $ds(P)$, is the set of all the possible derivatives of a component, P. The *current action type set* of a component P, $\mathcal{A}(P)$, contains all the action types (but not the rates) that are enabled in the current derivative of the component P. The *complete action type set* of a component P, $\vec{\mathcal{A}}(P)$, contains all the action types (but not the rates) that are enabled in any of the derivatives $P' \in ds(P)$, hence $\vec{\mathcal{A}}(P) = \bigcup_{P' \in ds(P)} \mathcal{A}(P')$. The *current action set* of a component P, $\mathcal{A}ct(P)$, contains all the actions (type and rate pairs) that are enabled in the current derivative of the component P. In addition it is necessary to construct a number of additional definitions.

Definition 2.1 (Fertile action)
An action γ is said to be fertile in derivative P_i if $P_i \xrightarrow{\gamma} P_j$ and $i \neq j$.

Definition 2.2 (Current fertile action type set)
The current fertile action type set of a component P, denoted $\mathcal{A}_f(P)$, is the set of all action types of actions that are fertile in the current derivative of P.

Definition 2.3 (Complete fertile action type set)
The complete fertile action type set of a component P, denoted $\vec{\mathcal{A}}_f(P)$, is the set of all action types of actions that are fertile in at least one derivative of P.

3 Behavioural independence and control

Put simply, *behavioural independence* requires that components in a model behave identically regardless of the current behaviour of other components in the model (this property is defined formally and discussed in detail in [9]). If a component is not behaviourally independent then it must be dependent on some other component to perform one of more actions during its evolution. This property is referred to as *control*, and is more formally defined thus:

Definition 3.1 (Control)
The component P is said to be subject to control in the model $P \underset{L}{\bowtie} Q$ if for any $P_i \in ds(P)$

$$\mathcal{A}ct\left((P_i \underset{L}{\bowtie} Q_j)/\{\mathcal{A}(P_i \underset{L}{\bowtie} Q_j)/\{\mathcal{A}_f(P_i) \cap L\}\}\right) \neq$$
$$\mathcal{A}ct\left((P_i \underset{L}{\bowtie} Q_k)/\{\mathcal{A}(P_i \underset{L}{\bowtie} Q_k)/\{\mathcal{A}_f(P_i) \cap L\}\}\right)$$

$\forall\, Q_j, Q_k \in ds(Q)$ *s.t.* $(P_i \underset{L}{\bowtie} Q_j), (P_i \underset{L}{\bowtie} Q_k) \in ds(P \underset{L}{\bowtie} Q)$

This definition states that the fertile actions in any derivative of P (or the rates at which those actions occur) can change as the current derivative of Q

evolves. This dependence is expressed by saying that component Q *controls* component P over action $K \subset L$ in $P \underset{L}{\bowtie} Q$ if the rate at which an action of type $k \in K$ can happen in $P_i \in ds(P)$ depends on the current derivative of Q. Clearly, if Q controls P over K then P cannot be behaviourally independent, but the independence, or otherwise, of Q is not given by this statement. The *control set* of P, $\mathcal{K}(P)$ is referred to as all the actions by which it is controlled. For example, if in the model $(P \underset{L}{\bowtie} Q) \underset{M}{\bowtie} R$, Q controls P over K_1 and R controls P over K_2 then $\mathcal{K}(P) = K_1 \cup K_2$. Clearly P is behaviourally independent iff $\mathcal{K}(P) = \emptyset$.

The trivial case for behavioural independence is clearly that with no shared actions, i.e. $P||Q$, however this is not the only case where components may be considered to be behaviourally independent. Furthermore, the fact that no actions are shared between two components does not mean they will always be behaviourally independent in the presence of other components. For example, in $(P||Q) \underset{L}{\bowtie} R$ the interaction between Q and R may influence the interaction between P and R, causing P and Q to be behaviourally *dependent*, i.e. Q may control R and R may control P.

The importance of the notion of control is two-fold. Firstly it allows where a model fails to be behaviourally independent to be explicitly stated. If the number of instances of control (the occurrence of actions through which control is exerted) is small, then it may be possible to create an approximate model where these instances are in some way ignored, or an approximate solution where the intervals between controls are long enough for the model to approach steady state behaviour. The second important use for the notion of control is in building approximations that actually seek to exploit this property. This topic is explored in the remaining sections of this paper.

3.1 Identifying control

Test cases have been developed for behavioural independence that capture many practical occurrences of that property at the component level, rather than the model level. For example, a component P will obviously be behaviourally independent in $P \underset{L}{\bowtie} Q$ if all actions $a \in L$ are defined in every derivative of Q at the same rate. Unfortunately the conditions for behavioural independence that can be applied at the component level are not complete, therefore the failure to identify behavioural independence does not imply control. In theory similar tests are possible for the notion of control, however, whilst it is a simple matter to show that a combination of components does indeed exhibit control, it is another matter to prove that the model can evolve into that combination of components without recourse to the global state space. Thus, even if $Q_i \overset{def}{=} (a, \alpha).Q_k$ and $Q_j \overset{def}{=} (a, \beta).Q_k$, the component P may still be behaviourally independent in $P \underset{L}{\bowtie} Q$ if there is no $P_i \in ds(P)$ such that both $P_i \underset{L}{\bowtie} Q_i \in ds(P \underset{L}{\bowtie} Q)$ and $P_i \underset{L}{\bowtie} Q_j \in ds(P \underset{L}{\bowtie} Q)$. The clear exception to this problem is where it can be shown that

1. Q can evolve into Q_i and Q_j using only actions that are not shared,
2. there is a $P_i \in ds(P)$ which may also be reached through internal actions,
3. a is defined passively (with rate $\top$) in P_i.

This first two of these conditions is necessary as only internal (not shared) actions can be assumed to happen, since any shared action may be blocked and it is only possible to know whether this is the case by exploring the derivatives of the model $P \underset{L}{\bowtie} Q$. The final condition, that a is defined passively in P_i, is necessary as a shared action may only be assumed to occur at a given symbolic rate only if one partner is passive or both partners specify the same symbolic rate.

Clearly this approach to identifying control is restrictive, indeed it is unlikely that any automated approach at the component level is able to identify 100% of cases. Furthermore, control is subject to numerical, as well as symbolic, conditions, thus if $\alpha = \beta$ then no control is exerted in this case. Clearly at the symbolic level of automated model inspection there is little that can be done to address numerical concerns, although it seems reasonable to assume that if the modeller has gone to the trouble of specifying different symbolic rates α and β for the same action type a, then there is a significant possibility that $\alpha \neq \beta$.

4 Approximation using reduced components

Given a model of two components, $X \underset{L}{\bowtie} Y$, such that X controls Y it is possible to generate X' such that the actions of Y are unaltered with respect to L in $X' \underset{L}{\bowtie} Y$ if for any derivative of $X \underset{L}{\bowtie} Y$ there is a corresponding derivative of $X' \underset{L}{\bowtie} Y$ such that

$$\mathcal{A}ct(X \underset{L}{\bowtie} Y) \cap L = \mathcal{A}ct(X' \underset{L}{\bowtie} Y) \cap L$$

In many situations, such as modelling resources, there will be little or no difference between X and the smallest possible X'. However, when X contains many derivatives, X_i where $\mathcal{A}ct(X_i) \cap L$ does not change, then there is potential for substantial state space reduction. The number of derivatives in X' is dependent on the number of actions in L and the number of distinct rates at which they are enabled. If the rates of each shared action is the same wherever it is enabled then the size of the reduced component X' is $2^{|L|}$. Thus the smaller the cooperation set the better the reduction in state space in the system $X' \underset{L}{\bowtie} Y$. Consider the case where $L = \{a\}$, if the rate of this action is the same in every derivative of $X \underset{\{a\}}{\bowtie} Y$ in which it is enabled then the reduced form of X, X' is given as follows.

$$\begin{aligned}
X'_1 &\stackrel{def}{=} (a, r1).X'_1 + (a, r2).X'_2 + (\tau, r3).X'_2 \\
X'_2 &\stackrel{def}{=} (\tau, r4).X'_1
\end{aligned}$$

The unknown action type τ is used to denote all those actions internal to X such that the derivative of X changes from X_i to X_j and $(\mathcal{A}ct(X_i) \cap L) \neq (\mathcal{A}ct(X_j) \cap L)$ (for all possible i and j. At this stage it is not possible to know the rates $r1$, $r2$, $r3$ and $r4$ in general, although the sum of the rates $r1$ and $r2$ is clearly the rate of a in every derivative of $X \underset{\{a\}}{\bowtie} Y$ in which it is enabled. If X is behaviourally independent in $X \underset{L}{\bowtie} Y$ then X may be solved in isolation and hence the rates found from the steady state probabilities. However, if Y controls X in $X \underset{L}{\bowtie} Y$ then it is not a simple matter to solve X. Instead an iterative approach is adopted (from [2]) to tackle this problem as follows.

1. Generate the smallest X' such that for any derivative of $X \underset{L}{\bowtie} Y$ there is a corresponding derivative of $X' \underset{L}{\bowtie} Y$ such that $\mathcal{A}ct(X \underset{L}{\bowtie} Y) \cap L = \mathcal{A}ct(X' \underset{L}{\bowtie} Y) \cap L$.
2. Similarly generate the smallest Y' such that for any derivative of $X \underset{L}{\bowtie} Y$ there is a corresponding derivative of $X \underset{L}{\bowtie} Y'$ such that $\mathcal{A}ct(X \underset{L}{\bowtie} Y) \cap L = \mathcal{A}ct(X \underset{L}{\bowtie} Y') \cap L$.
3. Solve $X' \underset{L}{\bowtie} Y$ using estimates of the unknown rates in X'.
4. Use the solution of $X' \underset{L}{\bowtie} Y$ to calculate the unknown rates in Y'.
5. Solve $X \underset{L}{\bowtie} Y'$ to find new estimates of the unknown rates in X'.
6. Solve $X' \underset{L}{\bowtie} Y$ to find new estimates of the unknown rates in Y'.
7. Repeat steps 5 and 6 until convergence (or abandon).

5 Example

Consider the following definition of a simple two queue network with feedback and blocking.

$$\begin{aligned}
Queue1_0 &\stackrel{def}{=} (arrival, \lambda).Queue1_1 + (feedback, \top).Queue1_1 \\
Queue1_j &\stackrel{def}{=} (arrival, \lambda).Queue1_{j+1} + (service1, \mu_1).Queue1_{j-1} \\
&\quad + (feedback, \top).Queue1_{j+1} \ , \ 1 \le j \le N-1 \\
Queue1_N &\stackrel{def}{=} (service1, \mu_1).Queue1_{N-1}
\end{aligned}$$

$$\begin{aligned}
Queue2_0 &\stackrel{def}{=} (service1, \top).Queue2_1 \\
Queue2_j &\stackrel{def}{=} (service1, \top).Queue2_{j+1} + (feedback, q\mu_2).Queue2_{j-1} \\
&\quad + (departure, (1-q)\mu_2).Queue2_{j-1} \ , \ 1 \le j \le N-1 \\
Queue2_N &\stackrel{def}{=} (feedback, q\mu_2).Queue2_{N-1} \\
&\quad + (departure, (1-q)\mu_2).Queue2_{N-1}
\end{aligned}$$

$$Queue1_0 \underset{\{service1,\, feedback\}}{\bowtie} Queue2_0$$

Each queue has an input action which is controlled by the other queue, i.e. it is blocked when the other queue is empty, $Queue1$ has an additional independent *arrival* process. Similarly each queue output action which is also controlled by the other queue, i.e. it is blocked when the other queue is full, $Queue2$ has an additional *departure* process. Hence, both components in this model ($Queue1$ and $Queue2$) are controlling and controlled over the co-operation set $\{service1, feedback\}$.

It is a simple matter to construct the reduced versions of $Queue1$ and $Queue2$, $Queue1'$ and $Queue2'$ respectively.

$$Queue1'_a \stackrel{def}{=} (arrival, \lambda).Queue1'_b + (feedback, \top).Queue1'_b$$

$$
\begin{aligned}
Queue1'_b &\stackrel{def}{=} (arrival, p_1\lambda).Queue1'_c + (service1, (1-p_2)\mu_1).Queue1'_b \\
&\quad + (service1, p_2\mu_1).Queue1'_a + (feedback, p_1 q\mu_2).Queue1'_c \\
&\quad + (feedback, (1-p_1)q\mu_2).Queue1'_b \\
Queue1'_c &\stackrel{def}{=} (service1, \mu_1).Queue1'_b
\end{aligned}
$$

$$
\begin{aligned}
Queue2'_a &\stackrel{def}{=} (service1, \top).Queue2'_b \\
Queue2'_b &\stackrel{def}{=} (service1, (1-p_3)\mu_1).Queue2'_b \\
&\quad + (service1, p_3\mu_1).Queue2'_c + (feedback, p_4 q\mu_2).Queue2'_a \\
&\quad + (feedback, (1-p_4)q\mu_2).Queue2'_b \\
&\quad + (departure, p_4(1-q)\mu_2).Queue2'_a \\
Queue2'_c &\stackrel{def}{=} (feedback, q\mu_2).Queue2'_b \\
&\quad + (departure, (1-q)\mu_2).Queue2'_b
\end{aligned}
$$

In this example the reduced form of the components have only three derivatives as there is no possible state where both the input and output actions are blocked. Clearly the derivatives $Queue1_a$, $Queue1_c$, $Queue2_a$ and $Queue2_c$, are analogous to each queue being either empty or full and the derivatives $Queue1_b$ and $Queue2_b$ correspond to all the non-empty and non-full conditions of each queue respectively. This approximation has introduced four new probabilities:

- p_1 is the probability that an additional job entering $Queue1$ will cause it to become full,
  $p_1 = Pr[Queue1_{N-1}]/(1 - Pr[Queue1_N] - Pr[Queue1_0])$.
- p_2 is the probability that a *service*1 action causes $Queue1$ to become empty,
  $p_2 = Pr[Queue1_1]/(1 - Pr[Queue1_N] - Pr[Queue1_0])$.
- p_3 is the probability that a *service*1 action causes $Queue2$ to become full,
  $p_3 = Pr[Queue2_{N-1}]/(1 - Pr[Queue2_N] - Pr[Queue2_0])$.
- p_4 is the probability that a job leaving $Queue2$ will cause it to become empty,
  $p_4 = Pr[Queue2_1]/(1 - Pr[Queue2_N] - Pr[Queue2_0])$.

The optimal values for these probabilities must be found by iteratively solving the two reduced models:

$$Queue1_0 \underset{\{service1,\ feedback\}}{\bowtie} Queue2'_a$$

and

$$Queue1'_a \underset{\{service1,\ feedback\}}{\bowtie} Queue2_0$$

Each of these reduced models has a CTMC with $3(N+1)$ states whereas the original model has a CTMC with $(N+1)^2$ states. Clearly there is a potentially significant saving on computation in each iteration, however the overall computational efficiency of this approach is dependent not only on the value of N, but the number of iterations required to achieve convergence over the introduced probabilities. Therefore in the following subsection the accuracy of the approximation and the number of iterations required are both investigated.

5.1 Numerical results

For the purposes of numerical evaluation the maximum sizes of the queues, N, was 9. This figure was chosen as a compromise between ease of solution of the complete model (for comparison) and significance of the model reduction. The complete model and the reduced (iterative solution) models were solved using the PEPA Workbench [1], to derive the generator matrices, and XMaple was used to solve these numerically. In all cases 20 iterations of the algorithm were used, however in most instances a good degree of convergence (6 decimal places) was achieved in 7 or 8 iterations. Two performance measures are derived for comparison, idleness and the average number of jobs in the system, and both these measures are compared with exact results computed directly from the complete model. Idleness, the percentage of time during which both queues are empty simultaneously, is calculated directly from the approximations as the average of $Pr[Queue1'_a \ \& \ Queue2_0]$ and $Pr[Queue1_0 \ \& \ Queue2'_a]$. It has been observed that when these two probabilities are very close to one another the estimated idleness is a very good approximation, and that the degree of separation between them is indicative of the degree of error. The second measure, the average number of jobs in the system, is calculated from the marginal queue size distributions derived from each of the approximations.

Figure 1 shows idleness plotted against arrival rate on a logarithmic scale. This plot clearly shows the extremely good approximation gained at low load, but the reduction in quality as the arrival rate increases.

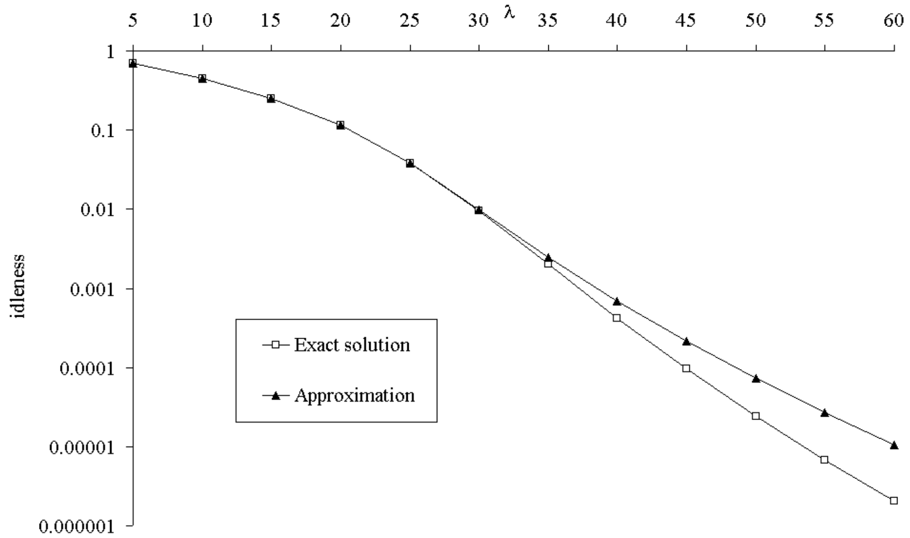

Figure 1: Idleness varied with arrival rate for exact solution and approximation $\mu_1 = \mu_2 = 60, q = 0.5$

The same input values are used in Figure 2, which shows the average number of jobs in the system. Whereas there was a significant error for the estimation of idleness, even at average load, there is no discernible error for the calculation of the average number of jobs. This is reiterated by Figure 3, which shows the percentage error from the data in Figure 2. Nowhere in this plot is the error worse than 0.5 percent. It is perhaps surprising that an approximate

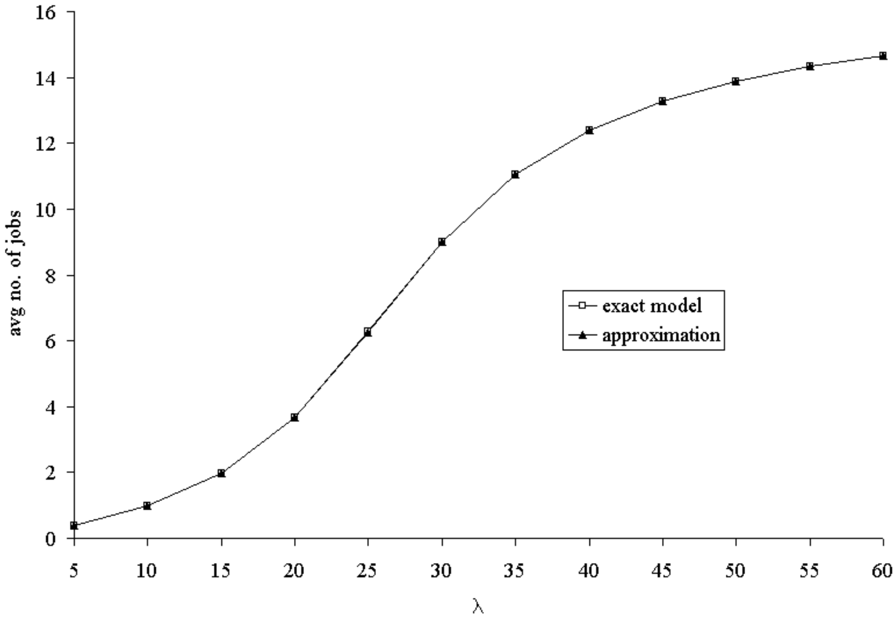

Figure 2: Average number of jobs varied with arrival rate
$\mu_1 = \mu_2 = 60, q = 0.5$

model (in this case two approximate models) should give rise to a very poor approximation when used directly but a very good approximation when using marginal probabilities. However, it is clear from the results presented here that, in this case at least, the errors introduced in the reduced models are almost entirely attributed to the approximated queues. Therefore that part of the reduced models which is modelled completely is actually behaving in an almost exact manner despite a very crude approximation to the rest of the system.

6 Conclusions and further work

In this paper we have presented an iterative approximation technique that can be applied to models in which all the components exhibit the property of control. Although the example presented here uses only two simple components, the technique is also applicable to multiple compound components. In addition the heart of this technique could easily be applied to cases where one component is behaviourally independent but controls another component. In the example the accuracy of the approximation for calculating marginal component distributions is shown to be extremely good, although the approximations themselves give only variable accuracy when used directly to derive measures of interest.

A significant amount of work remains to be done to improve the use of the technique described in this paper. In particular in the conditions for the formation of the reduced component we refer to the derivatives of the complete model $X \underset{L}{\bowtie} Y$. Clearly if the model is large then any reference to its global state space is highly undesirable, and if the model is not large then it is probably not necessary to apply model reduction techniques. Therefore it will be desirable to

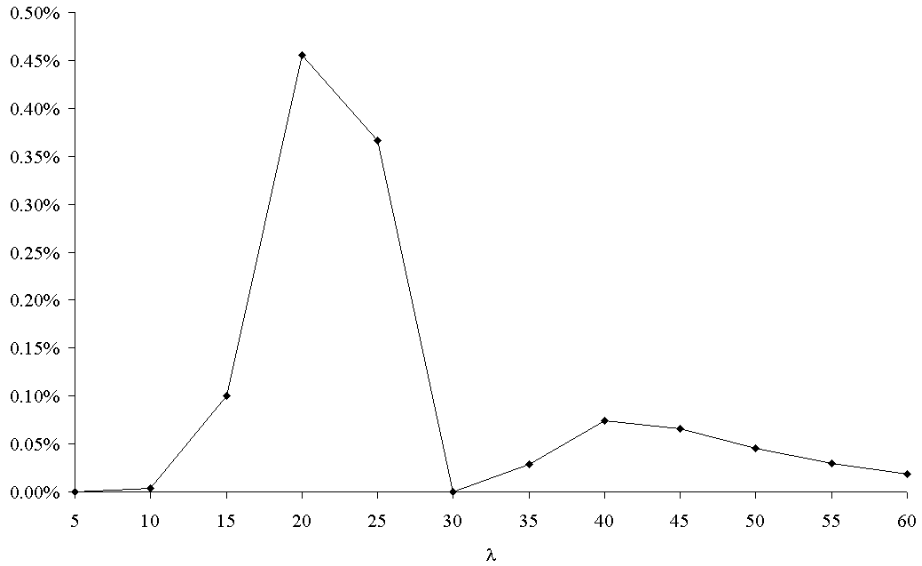

Figure 3: Percentage error of approximation of the average number of jobs varied with arrival rate

$$\mu_1 = \mu_2 = 60, q = 0.5$$

form conditions for the reduced component that do not rely on the global state space. The example presented does not satisfy the very restrictive condition developed in Section 3.1 to test for control and relies instead on the intuition of the modeller to identify the suitability of this approximation.

The approach taken so far in attempting to automate the construction of reduced components has been pragmatic. If no behavioural independence is identified then the ideal form for a reduced component is attempted at the component level. The global state space is explored only if it is not possible to form such a component and derive the necessary rates. In theory it should be possible to examine the model in advance to determine whether or not the global state space needs to be explored, however this has not yet been achieved. In addition, the conditions for convergence have yet to be explored. Although this has not been a practical issue it would be desirable to prove that a class of models will achieve convergence under this technique and to develop heuristics to predict the probabilities that are required to converge. Work also remains to be done to investigate if the approximations can be improved, perhaps for certain subclasses of model, particularly for measures (such as idleness in the example presented here) taken directly from the approximate model rather than those derived from the marginal distributions.

A further line of investigation concerns a slightly more flexible condition we have termed, *weak control*. A component Q is said to *weakly control* component P over $K \subset L$ in the model $P \underset{L}{\bowtie} Q$ if the occurrence of an action a, $a \in K$, affects only the rate of subsequent activities in P but not their existence, i.e. $\mathcal{A}(P)$ does not change as Q evolves, but the rates of its actions can. In this instance a reduced model of the kind presented in Section 4 could be constructed with average rates for actions in Q, rather than a different derivative of the reduced component Q' for every different rate of a. This can be thought of as analogous to the queueing scenario of approximating Markov modulated arrivals with a Poisson stream. Clearly the accuracy of this kind of approximation is likely to be much less predictable than that presented in this paper, however the class of applicable model is likely to be significantly larger.

References

[1] G. Clark, S. Gilmore, J. Hillston and N. Thomas, Experiences with the PEPA performance modelling tools, *IEE Proceedings - Software*, **146**(1), 1999.

[2] M. Gribaudo and M. Sereno, Approximation technique of finite queueing networks exploiting Petri net analysis, in: D. Kouvatsos (ed.), *Proceedings of the Fourth International Workshop on Queueing Networks with Finite Capacity*, Ilkley, UK, 2000.

[3] P.G. Harrison and J. Hillston, Exploiting Quasi-reversible Structures in Markovian Process Algebra Models, *The Computer Journal*, **38**(7), pp. 510–520, 1995.

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

[5] J. Hillston, Exploiting Structure in Solution: Decomposing Composed Models, in: C. Priami (ed.), *Proceedings of 6th International Workshop on Process Algebra and Performance Modelling*, 1998.

[6] J. Hillston, Exploiting Structure in Solution: Decomposing Composed Models, in: *FMPA Lecture Notes*, Springer-Verlag, 2001.

[7] J. Hillston and N. Thomas, A Syntactic Analysis of Reversible PEPA Models, in: *Proceedings of the Sixth International Workshop on Process Algebra and Performance Modelling*, 1998.

[8] N. Thomas, Behavioural Independence and Control in Markovian Process Algebra, Technical Report, Department of Computer Science, University of Durham, 2002.

[9] N. Thomas, Exploiting behavioural independence and control in Markovian process algebra, *submitted for publication.*