Bounded Decomposition of Stochastic Models

David Daly and William H. Sanders
Department of Electrical and Computer Engineering and
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1308 W. Main St., Urbana, IL, U.S.A.
{ddaly, whs}@crhc.uiuc.edu
http://www.crhc.uiuc.edu/PERFORM

***Abstract*— In this extended abstract we discuss bounding techniques for a collection of interacting simple QN and SAN models that we have been developing. Each simple model produces bounded output when presented with bounded inputs. A collection of bounded models can be used to construct larger models that have the same bounding properties. Additionally, the constructed model can be decomposed at the boundaries of the sub-structures, resulting in bounded submodels that can exchange bounded results. These bounds explicitly demonstrate the accuracy of the solution, and can be used to determine if the models need to be refined to increase the accuracy of the solution.**

This work is based on the observation that many simple Petri net structures can be compared to G/GI/1 queues either directly or indirectly. Bounds exist for total delay of customers through G/GI/1 queues, and for other quantities. We have developed new bounds for the departure process based upon existing bounds.

I. Introduction

A challenge to the use of state-based Markovian modeling for practical systems is the size of the generated state space. A great deal of work has been done on state space lumping to reduce the size of the state space, and in largeness tolerance (e.g., Kronecker, matrix diagram representations) to allow larger state spaces to be solved. The combination of the two techniques has increased the number of states that can be solved by several orders of magnitude. However, since the size of the state space of performance models grows exponentially, there are still many real-world performance, dependability, and performability models that cannot be solved, due either to the inherent size of the model, or to the need to use phase-type distributions (or more complicated processes) to accurately reflect the performance of the system.

In this paper we address the problem with a decomposition technique based upon bounding submodels. The decomposition of a large model into enough smaller submodels will result in models that can be solved using existing techniques. In our technique, the results of the submodels are monotonic with respect to the results received from other models, and the overall system is solved using a fixed-point iteration.

This material is based upon work supported in part by the National Science Foundation under Grant No. 9975019. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

The inputs to many simple Petri net and QN structures are arrival processes of tokens or jobs, and the outputs are departure processes of tokens or jobs. Therefore, it is appropriate to bound the inter-departure times based upon bounds on the inter-arrival times, or to bound departure times based upon bounds on the arrival times. More can be determined about the model from inter-departure time bounds, but the easier-to-prove departure time bounds are also useful.

We begin by analyzing a G/GI/1 queue, since many PN constructs can be analyzed in terms of a G/GI/1 queue. We have developed and extended a set of useful bounding properties for G/GI/1 queues, and include those results. We then discuss how the properties can be applied to several stochastic activity network structures, and the implications of these properties. Armed with a collection of model structures, we determine which results are useful to exchange, and how to solve for those results, and we discuss how the properties and models can be used in a fixed-point solution.

II. Partial Orders For Random Variables

We briefly review two partial orders for random variables used in this paper. The two partial orders for random variables are "usual stochastic order" and "increasing convex order." We say a random variable X is less than a random variable Y in the *usual stochastic order* ($X \leq_{st} Y$) if $P(X > a) \leq P(Y > a), \forall\ a$ [1]. $X \leq_{st} Y$ means that the inverse cumulative distribution function for X will be less than Y for all values, and thus the mean of X will be less than or equal to the mean of Y. The set of functions that preserve stochastic order are the set of all non-decreasing real functions.

A random variable X is less than a random variable Y in *increasing convex order* ($X \leq_c Y$) if $E[(X-x)^+] \leq E[(Y-x)^+], \forall x$, where $x^+ = max(x, 0)$ [1]. This order is preserved by the set of non-decreasing convex functions.

III. G/GI/1 Bounds

We begin by studying a */GI/1 queue. We use the following variables:

- A_i: the i^{th} arrival time
- a_i: the i^{th} inter-arrival time $A_i - A_{i-1}$
- D_i: the i^{th} departure time
- d_i: the i^{th} inter-departure time $D_i - D_{i-1}$
- S_i: the i^{th} service time

Further, we set $A_0 = a_0 = D_0 = d_0 = 0$. We generally assume that the service times are independent of all other variables, except other service times; however, there are conditions that allow the relaxation of this assumption.

A. Monotonicities

D_n can be expressed recursively in terms of arrival times and service times.

$$D_n = max(A_n + S_n, D_{n-1} + S_n) \tag{1}$$

Either the n^{th} arrival occurs after the $(n-1)$ departure, and the queue is empty, or the n^{th} arrival finds a non-empty queue. If it finds an empty queue, the n^{th} request will begin service immediately after arrival; otherwise, it will receive service after the previous request departs. Based on equation 1, it can be shown that D_n can also be written as:

$$D_n = max_{k \leq n}(\sum_{i=1}^{k} a_i + \sum_{i=k}^{n} S_i) \tag{2}$$

We study the queuing system with arbitrary sequence of service times (S_n), with two different sequences of arbitrary arrival times $(A_n) \leq (A'_n)$[1]. The arrival times are associated with the sequences of departure times (D_n) and (D'_n) respectively.

We note that $D_n = max(A_n + S_n, D_{n-1} + S_n)$, and max is a convex and non-decreasing function. Because max is non-decreasing, if the vector $(A_1, \ldots, A_n) \leq_{st} (A'_1, \ldots, A'_n)$, then $D_n \leq_{st} D'_n$ and $(D_1, \ldots, D_n) \leq_{st} (D'_1, \ldots, D'_n)$ for a G/GI/1 queue. Additionally, since max is convex and non-decreasing, if $(A_1, \ldots, A_n) \leq_c (A'_1, \ldots, A'_n)$, then $D_n \leq_c D'_n$ and $(D_1, \ldots, D_n) \leq_c (D'_1, \ldots, D'_n)$.

Additionally, D_n is a non-decreasing and convex function of the service times. Therefore, increasing the service times in the usual stochastic order (or increasing convex order) will lead to larger departure times in the usual stochastic order (or increasing convex order), and we can bound the departure process for a G/GI/1 queue when we use a service time distribution that is easier to analyze, but bounds the original service time process.

The analysis is more complicated for inter-departure times. We study the queue with two different sets of arbitrary inter-arrival times $(a_n) \leq (a'_n)$ associated with inter-departure times (d_n) and (d'_n) respectively. If $A_n < D_{n-1}$, then $d_n = S_n$, which is the minimal value for d_n. Therefore, $d'_n \geq d_n$. Otherwise, if $A_n > D_{n-1}$, it can be shown that $A'_n > D'_{n-1}$, and $(a'_1, \ldots, a'_n) \geq (a_1, \ldots, a_n) \Rightarrow d'_n \geq d_n$, as well as $(d'_1, \ldots, d'_n) \geq (d_1, \ldots, d_n)$, based on Eq. 2. It follows then that for a G/GI/1 queue, if $(a'_1, \ldots, a'_n) \geq_{st} (a_1, \ldots, a_n)$, then $d'_n \geq_{st} d_n$, and further that $(D'_1, \ldots, d'_n) \geq_{st} (d_1, \ldots, d_n)$.

[1]A sequence (X_n) is greater than or equal to a sequence (Y_n) ($(X_n) \geq (Y_n)$) if $X_i \geq Y_i, \forall i$. For the usual stochastic order, we say a random vector $X \geq_{st} Y$ if there exist random vectors $X' =_{st} X$ and $Y' =_{st} Y$, such that $x'_i \geq y'_i$ for all i and for all events in the event space.

B. Bounded measures

The preceding sections have shown that G/GI/1 queues preserve bounds on departure times and inter-departure times. While this property is useful, often the measure of interest will be of a different form.

In general, any non-decreasing function of the departure time (or inter-departure time) will be bounded based upon bounded arrival time (or inter-arrival times). For the arrival time bound, this implies that throughput is also bounded, since the throughput can be written as n/D_n as $n \to \infty$. Departure time bounds can also be useful in dependability, performance, and performability models in which the measure of interest is a first (or nth) passing time, or a related measure (e.g., reliability at time t).

For models that preserve the inter-arrival time bounds, even more measures are bounded. The delay experienced by requests to a queue and the queue length distribution are bounded by the inter-arrival time bound, shown in [1]. Additionally, all non-decreasing functions of the queue length, such as moments or the complete distribution, are bounded.

IV. Merging and Synchronizing Streams

Models commonly have multiple output streams from queues or other structures merging into one stream or waiting to synchronize with departures from other streams. Both merging and synchronization preserve departure time bounds, but do not preserve inter-departure bounds without added restrictions or modifications.

A. Synchronization

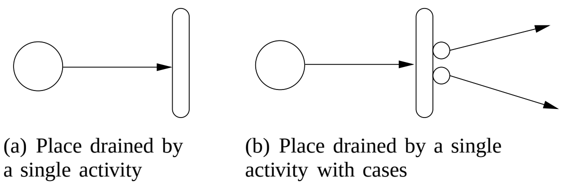
The departure time from a synchronization is the maximum of the time for a new arrival from each input. Since max is convex and non-decreasing, the synchronization preserves the departure time monotonicity properties of the G/GI/1 queue.

However, the inter-arrival times are not strictly the maximum of the inter-arrival values. As a result, it can be shown that the synchronization of two streams is not monotonic in the inter-arrival time of either stream.

Even if the model is simplified to a simple fork/join model, the inter-departure time will not be bounded, due to different processing times and queuing in the two (or more) branches. However, the inter-departure times will be bounded if the fork blocks until the previous fork synchronizes, ensuring that there is no queuing and waiting for resources in the parallel paths.

If the model cannot be constrained to be of the above form, we can construct approximate models that do bound the inter-departure times, provided that both arrival processes are renewal processes. However, unlike the previous models, if this model is provided with exact inter-arrival times, the result will not be an exact result, but will still be a bound.

The general problem with the synchronization structure was that an increase in one inter-departure time could be compensated for by a decrease in another inter-departure time. We can avoid this situation by changing the state of one or more of the processes when a synchronization occurs. When a synchronization does occur, at least one of the streams has


(a) Place drained by a single activity

(b) Place drained by a single activity with cases

Fig. 1. Simple bounded SAN structures

an arrival waiting at the synchronization. We alter the state of all such arrival streams to be the state with the stochastically largest (or smallest) delay distribution. For most (but not all) phase types this can be achieved by restarting the process (or putting it in its final phase).

B. Merging streams

A similar case is the merging of two or more streams. The departure time from a merge can be written as $D_n = min_{i:A_i^1>D_{n-1},j:A_j^2>D_{n-1}}(A_i^1, A_j^2)$. The min function is non-decreasing. Therefore, if the arrival rate for either stream is stochastically increased, the arrival rate to the merged stream will also be stochastically increased.

As in the case of synchronization, the inter-departure time does not preserve the usual stochastic order of the inter-arrival processes. The bounds hold only if a fork or choice feeds the merge, and is enabled only when all jobs or tokens have gone through the merge.

If the model cannot be constrained to be of the above form, we cannot conclude that bounds on the inter-arrival times will lead to bounds on the inter-departure times. However, as in the case of synchronization, we can construct approximate models that do bound the inter-departure times, provided that both arrival processes are renewal processes.

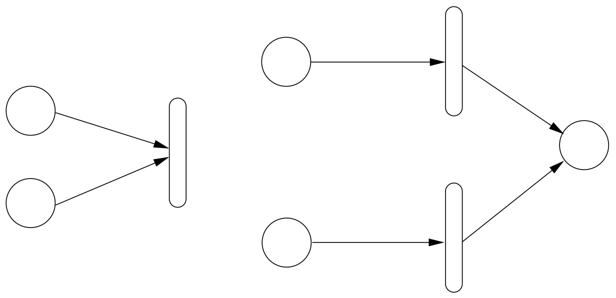
V. Stochastic Activity Networks

Based upon the G/GI/1 queuing result from the previous section, we now show monotonicity results for a number of common stochastic activity network (SAN) structures. These structures can be combined to form more complex models that still exhibit monotonicity.

Constructs that preserve departure time bound and inter-departure time bounds include a place drained by a single activity or a single activity with cases (Fig. 1), and a model that has arc cardinalities. Any one of those is is either a direct application of a G/GI/1 or a simple extension.

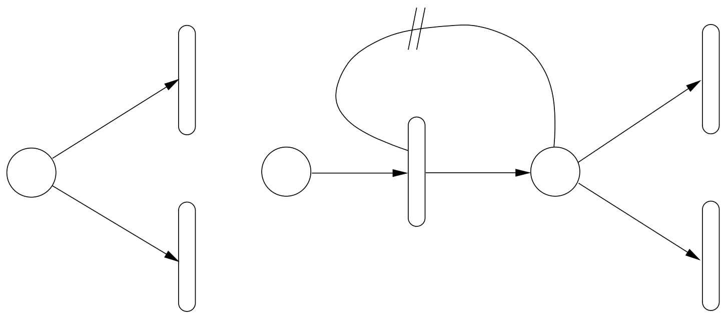
Two places drained by the same activity, and two activities feeding the same place (Fig. 2), are the SAN equivalents of synchronization and merging, respectively, and can be handled in the same manner. They will preserve departure time bounds and can be adjusted to deliver inter-departure time bounds.

The first model in Fig. 3 shows a place being drained by two competing activities. The two activities are in conflict, but share their sole enabling place, and the conflict is therefore a free choice. If both activities have exponential delays, this model is equivalent to the model using cases in Fig. 1, and will preserve departure and inter-departure time stochastic bounds.


(a) SAN synchronization

(b) SAN merging of streams

Fig. 2. Synchronization and merging in SANs


(a) Place drained by two competing activities

(b) Place drained by two competing activities, one token at a time[3]

Fig. 3. Two SAN models of competition

If either of the activities is not exponentially distributed, that equivalence fails. Then we must consider execution policy. Consider the case where both activities resample when either one completes. In that case, the two activities compete in the same manner for each and every token in the place, and the probability of a token going to either activity is the same for every token. This ensures that the departure times and inter-departure times will be monotonic in the inter-arrival times. The condition that each token is competed for in the same manner is also ensured if the model is adjusted to be of the form of the second model in Fig. 3.

Free choice was developed for untimed Petri nets. It has been described in this way: “Since, in such nets, one activity out of several activities involved in a conflict may be chosen freely and independently to fire, they are called *free choice nets*” [2]. In a timed model, the choice can be determined by past behavior, and could not be considered a free choice. The restriction in the second model in Figure 3 ensures that each token is competed for independently of each other. Therefore, we call the choice in the model a *timed free choice*. All timed free choice networks are bounded for departure time.

VI. Departure Process Representations

In order to use the bounded models described in the previous section, we need to determine bounded results to exchange. We

[3]The two hash marks indicate an inhibitor arc in a SAN

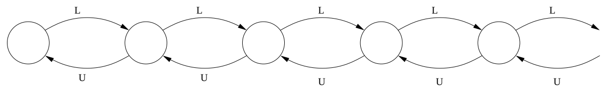


Fig. 4. Markov chain for an M/M/1 queue

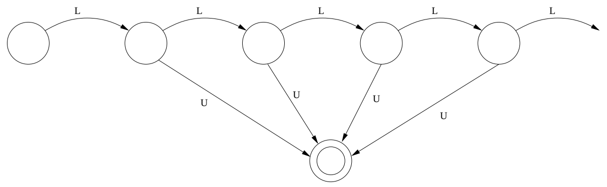


Fig. 5. Phase-type representation of departures from M/M/1 queue

want to represent the departure process of a subnetwork, and specifically to bound the inter-departure time of the process.

A renewal process is a simple result that can be used to bound the inter-departure times of a model. The use of a renewal process simplifies the analysis, as dependence between inter-departures times does not need to be considered. The cost of this simplification is that all inter-departure times will have the same delay distribution, and the nature of the process may make that representation inaccurate.

Regardless of the tightness of the approximation, we can still ensure that the end results are bounded, since the renewal bound will bound the inter-departure times. Therefore, we may use the results, and, if the result bounds are too loose, investigate more accurate bounds.

A. Generating renewal process bounds

To determine the renewal process bounds, we analyze the state space of the process. We create a phase-type distribution of the time between departures from the submodel by adjusting the model so that all activities representing departures go to an absorbing state. The adjusted model is a phase-type representation with multiple possible initial states (all the states originally reachable by a departure activity), and about the same number of states as the original model. However, because we have altered the form of the model, certain states may have become lumpable, and we can apply an optimal lumping algorithm to the resulting state space.

We use an $M/M/1$ queue as a quick example. The state space of the model is initially infinite, and all of the states can be reached by a departure, as shown in Fig. 4. We change all the departure activities to activities to an absorbing state, shown in Fig. 5. When we lump the model, we end up with a model with three states, corresponding to an empty queue, a non-empty queue, and the absorbing state. Once we modified the model, all the states with at least one job in the queue became strongly lumpable. The lumped departure process is shown in Fig. 6.

Once the lumping is performed, we must determine the stochastically largest and shortest delays possible based upon choice of initial state. In the case of the $M/M/1$ queue we can do so simply by inspection (as we can also do for $PH/PH/1$ queues). If all of the phase types are of the same basic type,

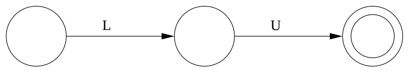


Fig. 6. Lumped phase-type representation of departures from M/M/1 queue

comparisons can be made on a phase-by-phase basis. However, the general case is much more complex due to the comparison of arbitrary phase type distributions, as well as to the fact that the phase types do not have to be stochastically ordered (for instance, one branch may be more likely to fire before time t_1 than another, but less likely at time t_2).

To handle a case in which the fastest or slowest phase type cannot easily be determined, we can create a phase type that is faster (or slower) than all of the others by enabling all the phase types and waiting for the first departure (or synchronizing on all the paths). Alternatively, we can explicitly solve and generate the delay distribution for each of the phase types. It is then trivial to take the maximum or minimum of all the distributions to make the the composite distribution. The composite distribution can then be fitted by a phase-type distribution using algorithms such as the EM algorithm [3] (the fitting will be approximate, but can be very accurate).

VII. Solving The Overall Model

The overall model can be solved via decomposition using the bounds and results described in this paper, if the model is too large to solve through conventional techniques. The overall model is decomposed into bounded submodels, which interact by exchanging results. The overall model is solved using fixed-point iteration. A submodel may need to be solved before other submodels that provide results to it, in which case loose inherent bounds are used to start the iterations. The monotonic property of the submodels ensures that the solution of every submodel leads to bounded results, since each submodel is solved using bounded inputs. Additionally, by comparing each iteration's results with the previous iteration's results, we can guarantee that the bounds will not diverge.

Decomposing a model into smaller submodels drastically reduces the size of the state space that needs to be solved at any one time, facilitating the solution of more complex and more detailed models. While decomposition has been used in the past to solve large models, in most cases, it has been used only for specific classes of models (e.g., product form), or there has been no guarantee on the accuracy of the solution. When bounding submodels are used, the accuracy of the solution is explicitly shown in the width of the bounds. Therefore, we are able to greatly increase the size of performance, dependability, or performability models that can be reliably solved.

References

[1] A. Muller and D. Stoyan, *Comparison Methods for Stochastic Models and Risks*, ser. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, 2002.

[2] W. Reisig, *Petri Nets: An Introduction*, ser. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1985, ch. 7, p. 101.

[3] S. Asmussen, O. Nerman, and M. Olsson, "Fitting phase-type distributions via the em algorithm," *Scandinavian Journal of Statistics*, vol. 23, pp. 419–441, 1996.