"'HOW FAR ARE WE FROM STEADY STATE?' ON-LINE ERROR BOUNDS FOR STEADY STATE APPROXIMATIONS"

by

K. PAPANIKAS*
N.M. VAN DIJK**
L.N. VAN WASSENHOVE***
and
E. YÜCESAN****

93/52/TM

* Research Assistant, at INSEAD, Boulevard de Constance, Fontainebleau 77305 Cedex, France.

** Professor at Universiteit Van Amsterdam, The Netherlands.

*** Professor of Production and Operations Management at INSEAD, Boulevard de Constance, Fontainebleau 77305 Cedex, France.

**** Associate Professor of Production and Operations Management, at INSEAD, Boulevard de Constance, Fontainebleau 77305 Cedex, France.

Printed at INSEAD, Fontainebleau, France

" How far are we from steady state ? "

On-line error bounds for steady state approximations.

Kleanthis D. Papanikas[2], Nico M. Van Dijk[1],

Luk N. Van Wassenhove[2] and Enver Yücesan[2]

Abstract

By studying performance measures via reward structures, on-line error bounds are obtained by successive approximation. These bounds enable one to determine when to terminate computation with a guaranteed accuracy. Furthermore, they provide insight into steady-state convergence in practical situations. The method is tested for a number of simple capacitated queueing networks. The results obtained indicate that the method provides a practical tool for numerically approximating performance measures of queueing networks.

Keywords: Markov reward structures, steady-state, queueing networks

---

[1] University of Amsterdam (U. v. A.), Amsterdam, The Netherlands,

[2] INSEAD, Fontainebleau, France.

1. Introduction

Over the last decades the use of Markov chains has become generally accepted to model and evaluate a variety of practical situations. Particularly, the use of Markov chains in queueing networks has become highly popular within such major application areas as manufacturing, telecommunications, and computer networks.

Typical performance measures of interest such as average throughput, average response time or system utilisation are to be seen as steady-state measures. Roughly speaking, these are measures that represent averages over a relatively long period of time. Many real systems, however, may have a finite "lifetime". They may therefore never reach steady state. A practical question is then how realistically the assumption of steady state reflects the actual situation. Or, relatively how long does it take for a system to reach a more or less stationary situation ?

Results from numerical analysis are available that show the convergence rate of the transition matrix iterations to its steady-state solution to be geometrically fast under a constraining constant depending on the second eigenvalue of this matrix [Cinlar 1975]. However, eigenvalues can only be obtained in more simplistic situations; their numerical calculations very rapidly become prohibitively expensive, if not impossible, for more realistic complex systems.

Unfortunately, no other type of results, even experimental ones, seem to be reported. Therefore, very little is known in general about how fast steady state is reached. In addition, analytic closed-form expressions, particularly product form expressions, are highly restricted when such practical considerations as finite constraints, breakdowns and job interference are taken into account [van Dijk 1993]. As a consequence, in such circumstances numerical computation of the performance measure of interest becomes attractive as an alternative to approximation methods or simulation. Indeed, the latter two procedures have the disadvantages of:

(i) being computationally expensive,

(ii) having no *a priori* guarantee of being unbiased, and

(iii) lacking a guaranteed error bound for accuracy.

A numerical approach in contrast also has to cope with the computational consequences (i) but it is usually guaranteed to be (asymptotically) correct. The remaining question of interest involves (iii) or alternatively whether a guarantee of accuracy can be provided when numerically approximating a performance measure. This latter question, in turn, also addresses the earlier question of how fast steady state is reached or at least how well a steady-state value of a performance measure is approximated.

The objective of this paper is therefore to provide a method to assess the accuracy of a steady-state value for a particular performance measure of interest. Such a method can be used for direct numerical computation. In addition, this approach can provide an indicator for the actual steady-state convergence rate in a realistic situation.

The key idea of our approach is to evaluate average performance measures by means of Markovian cumulative reward structures. A rather simple result, which has been developed and used in Markov decision theory, can then be adopted directly. This result was originally developed and used by Odoni [1969] and Popyack [1985] to determine the accuracy of successive approximation schemes in dynamic programming (also see Ross [1985] and Tijms [1986]). So far, however, it has not been applied for the purpose of approximating steady state performance measures.

To this end, a somewhat simplified result, adopted from the literature, will be presented for direct application to Markov reward chains. Next, its potential will be illustrated on some generic queueing networks. An illustrative set of numerical results is included. A more extensive study is undertaken in [Papanikas, 1992]. These results show that the error bounding technique can be used in a straightforward fashion to determine truncation points for computations and to guarantee a level of accuracy with respect to a steady-state value. As such, it can be regarded as a simple and guaranteed tool for realistic applications.

This paper is organized as follows: Section 2 presents the general approach and the result that provides on-line error bounds. In Section 3, the method is applied to a two-station assembly line structure. Section 4 provides a discussion on the operating characteritics of the proposed method. The discussion is summarized in Section 5.

2. Methodology

2.1. Preliminaries

Consider a continuous-time Markov chain (CTMC),{X(t): t>=0, (Ω, Q, r)}, with a state space Ω={ 0,1,2,3, $\cdots$ }, an infinitesimal generator matrix with transition rate q(i,j) from state i to state j, **Q** = [q(i,j)], and a one-step reward rate, r(i) when the system is in state i. It is further assumed that:

(i) The Markov process is irreducible at Ω with a unique steady-state distribution $\{ \pi(i) \}_{i \in \Omega}$.

(ii) Without loss of generality, a value $D < \infty$ exists for which

$$\sum_{j \neq i} q(i, j) \leq D \quad (i \in \Omega) \tag{2.1}$$

(iii) For a given reward rate, r(i), the measure **g** is well defined by

$$\mathbf{g} = \sum_{i} \pi(i) \cdot r(i) \tag{2.2}$$

The average expected reward of the model can be more easily obtained after the standard uniformization technique has been applied. The goal of using this technique is the prospect of studying our continuous-time model as a discrete-time Markov chain with a probabilistically equivalent generator matrix.

The uniformized one-step transition probability matrix **P** = [p(i,j)] is defined as :

$$p(i,j) = \begin{cases} q_{(i,j)}/D & (j \neq i) \\ 1 - \sum_{i \neq j} q_{(i,j)}/D & (j = i) \end{cases} \tag{2.3}$$

It can be directly seen that the generator $\mathbf{Q}^d$ of the discrete-time Markov chain over epochs {0, 1/D, 2/D,.....} with one-step transition matrix P coincides with the infinitesimal generator matrix Q=[q(i,j)] of the continuous-time model, as $\mathbf{Q}^d = (\mathbf{P} - \mathbf{I}) \cdot D = \mathbf{Q}$ where I is the identity matrix.

The fact that the two models have equivalent generator matrices implies that the steady-state distributions are also equal $\pi = \pi^d$. The latter are determined by $\Pi\, \mathbf{Q} = 0$ and $\pi^d\, \mathbf{Q}^d = 0$. As a consequence, related performance measures such as average expected reward g for a given reward rate r are also equal. We can thus restrict our attention to the discrete-time model.

In order to study the average reward for our discrete-time model and with $\mathbf{P}^k$, the k-th power of P, for each n = 0,1,2,3,.., we define the expected total reward functions V_n, by:

$$V_n(i) = \sum_{k=0}^{n-1} \mathbf{P}^k r = \sum_{k=0}^{n-1} \left\{ \sum_j p^k(i,j) \cdot r(j) \right\} \tag{2.4}$$

In words this expression represents the expected cumulative reward over n steps when starting in state i at time 0 and receiving a reward r(j) at a visit to state j.

Then, by virtue of assumptions (i) and (ii), we can conclude that, for an arbitrary initial state i, the following expression is well defined and represents the average expected reward per unit time:

$$\mathbf{g} = \lim \frac{1}{n} V_n(i), \quad n \to \infty \tag{2.5}$$

This average reward value represents accurately different performance measures of interest through the appropriate specification of the reward rate function r.

Example As an example consider a M/M/1/N loss system (that is, a single-server system with finite waiting room of size N-1 and exponential interarrival as well as service times) with arrival rate λ and service rate μ. The following measures in Table 1 can then be obtained, where $1_{\{A\}}$ represents the indicator of an event A; i.e., $1_{\{A\}} = 1$ if event A is satisfied and $1_{\{A\}} = 0$ otherwise.

g	r
Throughput	$\mu \cdot 1\{z > 0\}$
Loss probability	$1\{z = N\}$
Tail probability t	$1\{z > t\}$
Mean number	z

Table 1.

The following simple recursion relation, which is directly obtained from (2.4) and which is also known as a one-step Markov reward or dynamic programming relation,

$$V_n(i) = \begin{cases} 0 & \text{for } n = 0 \\ r(i) + \sum_j p_{(i,j)} V_{n-1}(j) & \text{for } n \geq 1. \end{cases} \tag{2.6}$$

provides a simple computational scheme, also known as successive approximation, to calculate or better approximate the value g, using (2.5) and the starting values:

$$V_0(i) = 0 \text{ for all i.}$$

In the next subsection, however, we will show that this same scheme can also be used to approximate g without averaging, that is (2.3), but in contrast by providing monotonically converging lower and upper bounds of g.

2.2. On-line error bounds

To this end, define values $\mathbf{M_n}$ and $\mathbf{m_n}$ by

$$\begin{aligned} \mathbf{m}_n &= \min_j [V_n(j) - V_{n-1}(j)] \\ \mathbf{M}_n &= \max_j [V_n(j) - V_{n-1}(j)] \end{aligned} \tag{2.7}$$

The following result then applies. This result is adopted from results in Markov decision theory (e.g. see Tijms [1986, Chapter 3]). In that setting, it has already been made explicit and proven by Odoni [1969], and further extended most notably by Popyack [1985]. In a non-decision setting, however, it seems to have remained unmentioned and unexploited. For the directness in the present setting and for self containment, we prefer to present a compact proof.

Result:

$$\mathbf{m}_n \leq \mathbf{m}_{n+1} \leq \mathbf{g} \leq \mathbf{M}_{n+1} \leq \mathbf{M}_n \qquad (\mathbf{n} \geq 0) \tag{2.8}$$

Proof First, note that, for all i, by (2.7)

$$\begin{aligned} V_n(i) &\geq V_{n-1}(i) + \mathbf{m}_n \\ V_n(i) &\leq V_{n-1}(i) + \mathbf{M}_n \end{aligned}$$

while by (2.6)

$$V_n(i) = r(i) + \sum_j p(i,j) V_{n-1}(j) \tag{2.9}$$

so that

$$\mathbf{m}_n + V_{n-1}(i) \leq r(i) + \mathbf{P}V_{n-1}(i) \leq \mathbf{M}_n + \mathbf{V}_{n-1}(i) \tag{2.10}$$

First, consider the left inequality of (2.10) and define the function δ by:

$$\delta(i) = [r(i) - \mathbf{m}_n] + \mathbf{P}V_{n-1}(i) - V_{n-1}(i) \qquad \text{(for all i)} \tag{2.11}$$

Further, for convenience, set $\mathbf{V} \equiv \mathbf{V}_{n-1}$ and for the scalar $\mathbf{m}_n$ let $\overline{\mathbf{m}}_n$ denote the function $\overline{\mathbf{m}}(i) = \mathbf{m}_n$ for all i. Then, in functional equation and for arbitrary N,

$$\begin{aligned} V &= \mathbf{P}V + [r - \mathbf{m}] - \delta \\ &= \mathbf{P}\{\mathbf{P}V + [r - \mathbf{m}]\} + [r - \mathbf{m}] - \delta \\ &\cdots\cdots \\ &\cdots\cdots \\ &= \mathbf{P}^k V + \sum_{k=0}^{N-1} \mathbf{P}^k r + \sum_{k=0}^{N-1} \mathbf{P}^k \overline{\mathbf{m}} - \sum_{k=0}^{N-1} \mathbf{P}^k \delta \end{aligned} \tag{2.12}$$

As the matrix $\mathbf{P}^k$ is a probability matrix (that is, non-negative and with row sums equal to 1), first note that, for all i,

$$\sum_{k=0}^{N-1} \mathbf{P}^k \overline{\mathbf{m}}(i) = N\overline{\mathbf{m}}$$

Furthermore,

$$\sum_{k=0}^{N} \mathbf{P}^k \delta(i) \geq 0 \tag{2.13}$$

where we used that $\delta(i) \geq 0$ for all i by definition (2.11) and inequality (2.10).

Next, divide both the left and right hand sides by N and let $N \to \infty$. As the function $\mathbf{V} = \mathbf{V}_{n-1}$ is a bounded function by the assumption that r is bounded, we then conclude, for any i:

$$\frac{1}{N} V(i) \longrightarrow 0 \quad (N \longrightarrow \infty)$$

$$\frac{1}{N} \mathbf{P}^N V(i) \longrightarrow 0 \quad (N \longrightarrow \infty) \tag{2.14}$$

By also recalling (2.5) with N substituted for n, we thus obtain, by combination of (2.12), (2.13) and (2.14), $0 \leq \mathbf{g} - \mathbf{m}$. Similarly, by considering the second inequality in (2.10), $0 \geq \mathbf{g} - \mathbf{M}$.

To prove the monotonicity, we conclude from (2.10) :

$$\begin{aligned} \mathbf{m}_{n+1} &= \min_i [V_{n+2}(i) - V_{n+1}(i)] \\ &= \min_i \sum_j p(i,j)[V_{n+1}(j) - V_n(j)] \geq \min_i \sum_j p(i,j)\mathbf{m}_n = \mathbf{m}_n \end{aligned}$$

and similarly $\mathbf{M}_{n+1} \leq \mathbf{M}_n$ is proven. This completes the proof of (2.8). ·

3. An illustration

In this section, we apply the approach outlined in Section 2 to a simple but nevertheless unsolvable and illustrative example: a finite assembly line.

3.1. Model

We consider a simple assembly line structure, a tandem queue which consists of two single-server queues in series as depicted in Figure 1. Jobs arrive at the first queue according to a Poisson process with parameter λ. The service requirements are assumed to be exponentially distributed with parameters μ_1 and μ_2 at queues 1 and 2, respectively. Jobs are served in a first-come-first-served order. In addition, queue 1 has a capacity for at most N_1 jobs in total and queue 2 for at most N_2.

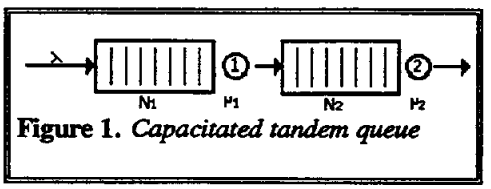

Figure 1. *Capacitated tandem queue*

A customer is denied access upon arrival when the first queue of the network is saturated (n1 = N1). A customer is recirculated to the first station if, upon its service completion at station 1, the second station is saturated (n2 = N2). Here we note that, because of the memoryless property of the exponential distribution, one can also state that the first station stops servicing when the second one is saturated. This, in turn, is known as *communication blocking* (Altiok and Perros [1986]).

The state of the system can be described by the vector $\bar{n} = (n_1, n_2)$ where n_i denotes the number of customers at station i, i=1,2. By $\bar{n} + e_i$ ($\bar{n} - e_i$), we denote the state of the system equal to $\bar{n}$ except for one customer more (less) at station i where $\bar{n} - e_i = \bar{n}$ for $n_i = 0$. Consequently, by $\bar{n} - e_i + e_j$ we denote the state equal to $\bar{n}$ with one customer moved from station i to station j, where i=0 corresponds to an external arrival at station j (j can only be the first station of the configuration) and j=0 to a departure from the system at station i (again i is restricted to be the second station in the configuration).

The transition rates can be summarised in the following way:

$$q(\bar{n},\bar{n}') = \begin{cases} \lambda \cdot 1_{\{n_1 < N_1\}} & \bar{n}' = \bar{n} + e_1 \\ \mu_2 \cdot 1_{\{n_2 > 0\}} & \bar{n}' = \bar{n} - e_2 \\ \mu_1 \cdot 1_{\{n_1 > 0\}} 1_{\{n_2 < N_2\}} & \bar{n}' = \bar{n} - e_1 + e_2 \end{cases}$$

Hence, with $D = \lambda + \mu_1 + \mu_2$, the uniformization constant, in accordance with (2.3), yields the probability of staying in the same state:

$$q(\bar{n},\bar{n}) = (1 - [\lambda \cdot 1_{\{n_1 < N_1\}} - \mu_2 \cdot 1_{\{n_2 > 0\}} - \mu_1 \cdot 1_{\{n_1 > 0\}} 1_{\{n_2 < N_2\}}]/D).$$

Two performance measures are used:

1. The throughput, T, of the system is defined as the number of customers departing from the second queue per unit of time. The reward rate, $r(\bar{n})$, in this case is given by:

$$r(n_1, n_2) = \begin{cases} \mu_2 & n_2 > 0 \\ 0 & n_2 = 0 \end{cases}$$

2. The expected sojourn time, W, is defined as the time a customer spends in the system, which includes both the service time and the waiting time in each queue. This sojourn time can be calculated through Little's law ($L = T \cdot W$), where L is the average number of customers in the system. To calculate L, in turn, one can use:

$$W = L / T.$$

3.2. Numerical results

A presentation of the system characteristics is given below. They are also summarised in Table 2.

- Different utilisation values, between 20% and 120%, have been considered.

- The capacity of each queue was taken to be equal to four customers. The effect of capacity on the convergence rate was also examined through subsequent modifications of this buffer size. The values used were always kept between the following limits: $1 \leq \text{capacity} \leq 30$.

- The performance measures used for the analysis are the throughput of the system and the sojourn time. The average number of customers in the system is also readily available, since its calculation was an essential step in the process of deriving the waiting times.

Characteristics of the system			
Number of queues	**Servers in each queue**	**Capacity**	**Utilisation**
2	1	1 to 30	20% to 120%

Table 2.

- The uniformization variable D and the corresponding step-size 1/D per transition was chosen to be

$$D = \lambda + \mu_1 + \mu_2 \tag{3.1}$$

- As stopping criterion we used:

$$M_n - m_n \leq \varepsilon \tag{3.2}$$

where $\varepsilon = 0.001$.

4. Sensitivity analysis and further discussion of numerical results

In this section we discuss the operating characteristics of the proposed method in detail. In particular, the following aspects will be elaborated upon:

- its performance (convergence speed),
- sensitivity with respect to the characteristics of the system under study (capacity, utilisation), and
- sensitivity with respect to its parameters (uniformization variable, stopping criterion).

Though these discussions are not directly necessary for an understanding and actual application of the method, we believe them to be of interest to practitioners to obtain further insight into some underlying features, some global observations, and some conclusions that can be drawn from the experiments. As such results are usually not reported (in detail) but are essential for practical usage, we find it useful to include them.

4.1 Performance of the method

A representative illustration of the numerical experiments is provided in Figures 2 (throughput) and 3 (sojourn time) for different utilization levels. The illustration pertains to a system consisting of two single-server queues in tandem, each with a waiting room for four customers. These figures indicate that the lower and upper bounds m_n and M_n can be used as simple practical values to quickly determine the value of g within a prescribed accuracy. As such they support the Markov reward method as an easy-to-use *practical tool.*

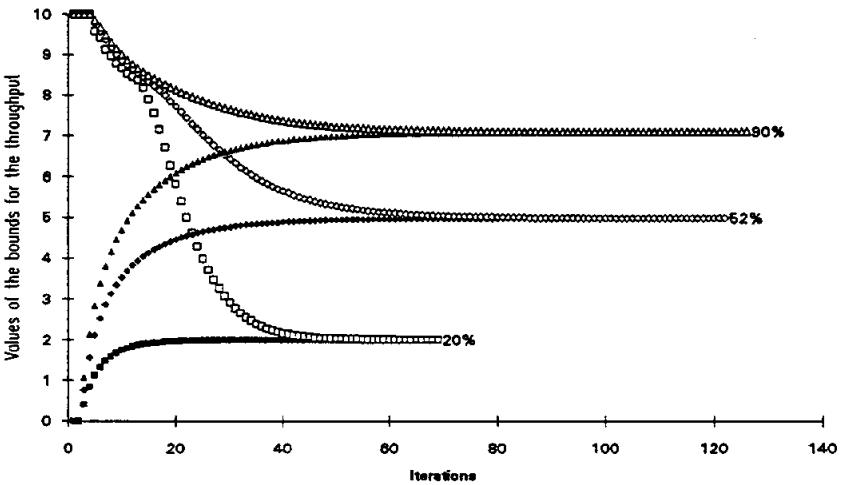

Figure 2. *Convergence rate for system throughput*

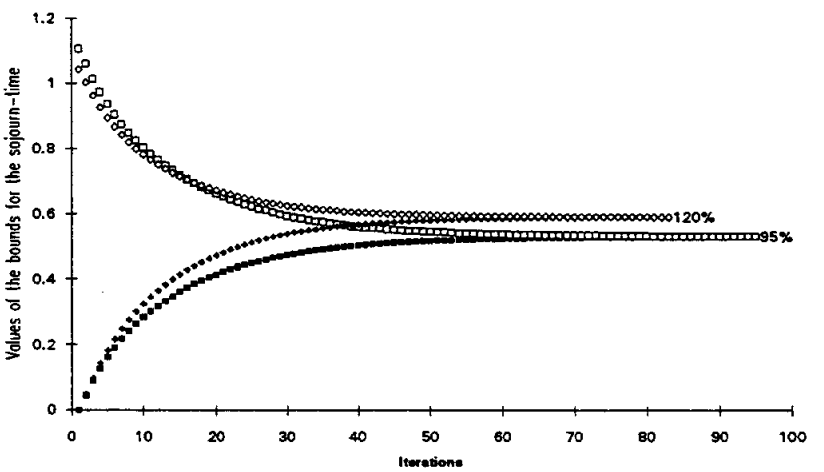

Figure 3. *Convergence rate for sojourn time*

In addition to the computational accuracy, the results also provide an indication of the actual convergence to a steady-state measure in terms of transitions (or events). In terms of the original (non-uniformized) CTMC $\{X(t) : t >= 0\}$, we can consider the transitions from state i to occur at a uniform rate of D rather than at a state-dependent rate $\Sigma q(i,j)$. However, only a fraction $\Sigma q(i,j) / D$

are real transitions out of state i corresponding to an event in the original queueing process such as an arrival or a service completion. The remainder are just fictitious transitions that leave the process in state i. As a consequence, the number of iterations required can be regarded as an upper bound on the number of events (arrivals or completions) that one has to observe in a real situation or in a discrete-event simulation before the prescribed accuracy for a steady-state performance value is achieved.

The method is observed to converge fairly quickly; however, its rate of convergence, to the best of our knowledge, has not yet been formally proved. In this section we examine this rate and more specifically we show that it actually follows the pattern of a geometric function :

$$F(x) = \alpha \cdot \beta^x$$

For convenience, the following transformation was performed :

$$F(x) = \alpha \cdot \beta^x \equiv F(x) = e^{a+b \cdot x} \Leftrightarrow Ln(y) = c + d \cdot x$$

where the model at the right is known as the exponential model with c = ln a and d = ln b.

We used the method of *least - squares estimation* for the purpose of *curve - fitting* because the data failed to satisfy the underlying assumptions of regression analysis. The method has been applied to different cases with different capacities and utilisation levels. Figure 4 shows, for the configuration with capacity 1 and 90 % utilisation, the differences of the upper bound for the value of the system throughput, as calculated through the program at every step, and the steady-state value of the throughput, computed by solving a set of differential - difference equations.

Two distinct curves are presented in Figure 4. The first one represents the differences as calculated by the program, while the second one corresponds to the values predicted through the least square estimates. In both cases the values are plotted against the number of iterations. Our results are consistent with the derivations in Cinlar [1975, p.378], where geometric convergence rates are established for irreducible, aperiodic Markov chains.

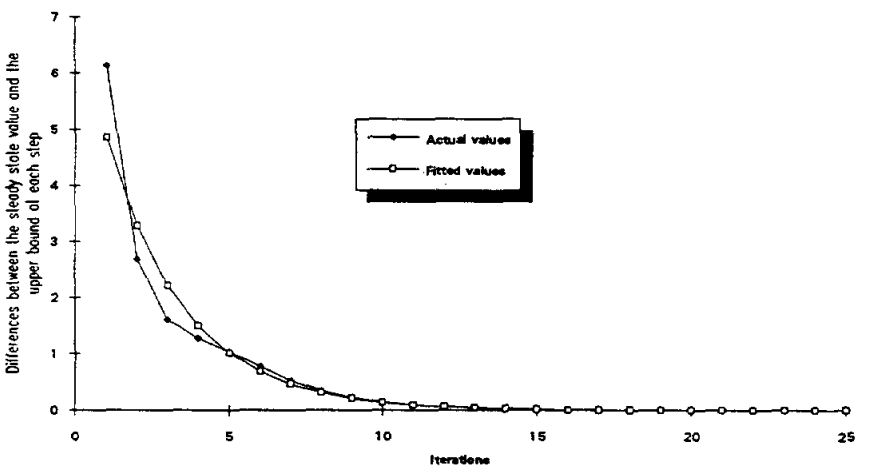

Figure 4. *Convergence rate of the algorithm*

4.2. Characteristics of the configuration

It has been empirically shown that the convergence rate of the algorithm is geometric. This result supports the algorithm's utility as a practical tool for the steady-state approximations of queueing network performance measures. Implications of the above result with respect to practical applications are discussed in this section.

An examination of the convergence rate with respect to changes in the system characteristics is certainly of great interest. Since the work in this paper has been restricted to relatively simple systems, we have only examined the impact of two system characteristics, utilisation and capacity. As anticipated, an increase in the value of these two parameters necessitates a larger number of steps before convergence. As depicted in Figure 5a, an increase in utilization modifies the distribution of the probability mass throughout the state space. For instance, at high utilization levels, most of the mass is concentrated around the congested states, which are attained very quickly. For low utilization rates, however, the opposite is true. As depicted in Figure 5b, convergence is slower when the probability mass is more evenly distributed over the state space; those corresponds to the range of 0.6 to 0.8.

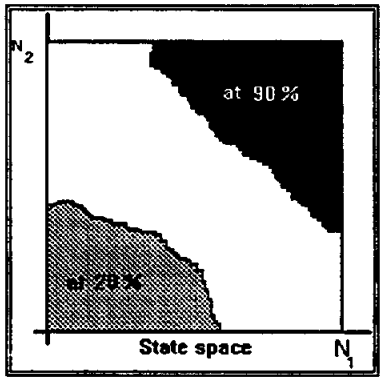

Figure 5a. *Distribution of the probability mass over the state space at different utilization levels.*

Increasing capacity results in an exponential growth of the number of iterations ($y = ae^{bx}$, where x is the capacity of each station). Such a growth can quickly lead to prohibitively large run-time requirements (Figure 6). This is also an intuitive result as an increase in capacity corresponds to an increase in the number of states. In general, a system with a larger state space will take longer to attain steady state.

The steep increase in the number of iterations for systems with larger state space appears to limit the applicability of the proposed method to complex systems. There are, however, two approaches to mitigate this problem: truncation of the state space and acceleration of the convergence rate. These approaches are discussed next.

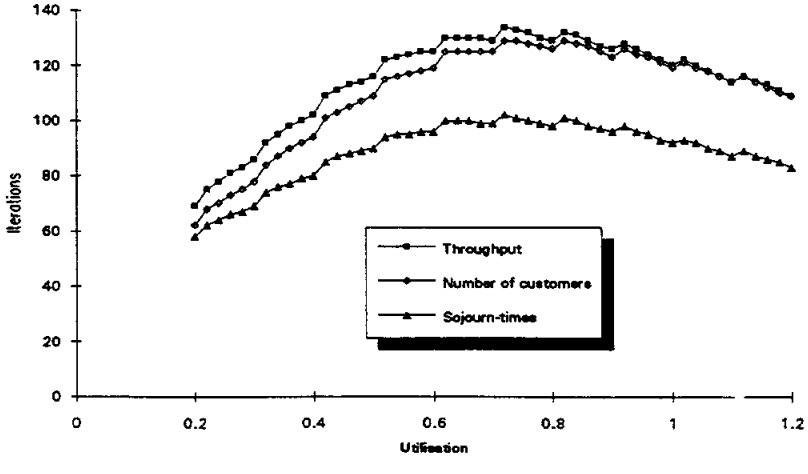

Figure 5b. The number of iterations versus utilization

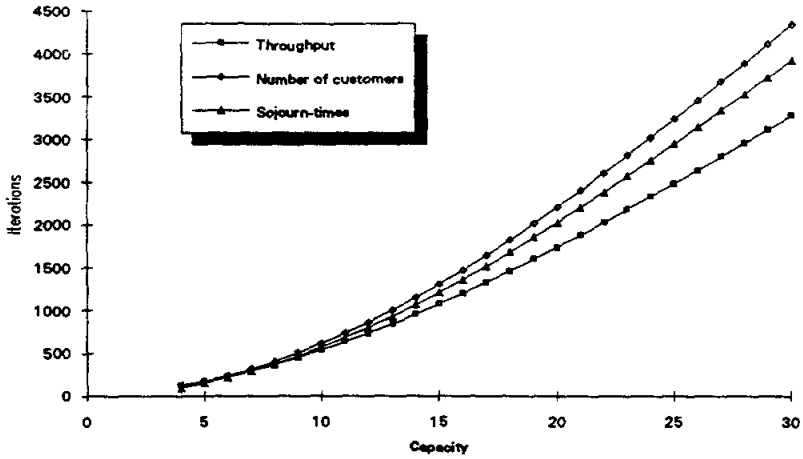

Figure 6. *The number of iterations versus system capacity*

4.3. Experimental truncation error bound

Examination of a system with a very large state space can prove to be a very difficult task . Thus, such a state space is normally truncated to be able to obtain a solution, but with no guarantee for the accuracy of the results. A question of great interest is then whether this truncation error could possibly be quantified.

Truncation errors have been experimentally obtained (for the configuration considered in section 3.1). The results indicate that a significant reduction in run times can be obtained by truncating the state space with a small truncation error. For instance, a truncation from 900 states (capacity of 30 at each station) down to 625 yields a truncation error in the values of the average throughput of 0.038 (0.43%). Table 3 shows the error in throughput at 90% utilization level due to the truncation of capacity from 30 customers at each station to smaller values.

Further analysis reveals that the capacity restriction imposed on our configuration hardly makes any difference, compared to the uncapacitated system, when the maximum allowed number of customers increases above 60. Thus, the existing *steady-state* equations of the equivalent *product-form* configuration could be used to compute the value of the system throughput with a negligible error. Papanikas [1992, p.47] discusses analytic methods to provide *a priori* error bounds in Markov chains.

Capacity	Throughput (average)	Difference	Percentage Δ%
10	8.385727	0.567223	6.33
15	8.699578	0.253372	2.83
20	8.842422	0.110528	1.23
25	8.914626	0.038324	0.43

Table 3.

4.4. An attempt to accelerate.

Tijms [1986, p.195] considers a relaxation method which could practically accelerate the convergence rate of the value-iteration algorithm. This modification, however, has no theoretical guarantee of convergence. It is based on a relaxation factor ω, which can be dynamically selected. The relaxed version of the algorithm computes, at the n-th step, a new approximation of the reward function $V_n(i)$ obtained by using both the previous values $V_{n-1}(i)$ and the residuals $V_n(i) - V_{n-1}(i)$.

The technique resembles the successive overrelaxation used for solving a system of linear equations. The objective is to force the difference between the upper and lower bounds to decrease quicker.

The outcome of our experiments with this modified version was not very impressive, since, in most of the cases, the number of iterations was reduced only by one. Nevertheless, the use of a relaxation factor could prove to be fruitful when the algorithm is applied to bigger systems, like queueing networks with a large number of stations. The differences in the number of iterations, with and without the relaxation factor are presented in Table 4.

Utilisation (%) :		30	50	70	90	120
Number of Iterations	normal	86	126	129	126	109
	relaxed	85	115	128	123	105

Table 4.

4.5. Characteristics of the algorithm

The convergence rate of the algorithm is also influenced by two of its parameters: the value of the uniformization variable (D), and the value of the stopping criterion, ε.

The increase in the number of iterations resulting from an increase in the value of D can be estimated through the following equation :

$$\frac{D}{Iterations} = \frac{D'}{Iterations'}.$$

As the best possible value for D, we recommend the one which satisfies equation (2.1) with equality.

The value of the stopping criterion has a big impact on convergence speed. A smaller value, which will provide a more accurate approximation, will result in a larger number of iterations.

	Utilisation (%)			
D	20	50	90	120
32	103	149	139	109
40	130	188	174	137
ε				
0.001	69	116	126	109
0.0001	82	143	157	137

Table 5.

An appreciation of the changes in convergence rate due to changes in the values of D and ε can be obtained from Table 5.

5. Evaluation

The issue of whether or not a complex stochastic system, such as those frequently arising in manufacturing, telecommunications and computer networks, has reached steady state is of practical interest to justify the use of computational evaluation methods or even explicit expressions based on the steady-state assumption. This question, in turn, is directly related to the question of how accurately a steady-state performance measure is approximated when employing a numerical computation.

To this end, a computational method is suggested to calculate the values of performance measures by successive Markov reward approximations. This method enables one to obtain on-line error bounds for the accuracy of the approximation. Extensive numerical analysis conducted in [Papanikas, 1992] and summarized in this paper illustrates that these bounds can be quite practical to provide a guaranteed accuracy within a reasonably small number of steps: as such, the results support the approach as a practical tool. The bounds could also be used to show how the convergence rates depend on system input parameters, such as capacity and load.

Acknowledgement

The authors gratefully acknowledge the support of INSEAD through R&D Project No 2264R.

References

[1]. **Altiok and Perros**, (1986), *Open networks of queues with blocking : split and merge configurations*, **IIE Transactions**.

[2]. **Cinlar, E.** (1975), *Introduction to Stochastic Processes*, Prentice Hall. Englewood Cliffs, NJ.

[3]. **Odoni, A. R.** (1969), *On finding the maximal gain for Markov decision processes*, **Operations Res. 17 (5)**, 857 - 860.

[4]. **Papanikas, K.** (1992), *On-line error bounds for steady state approximations,* Unpublished MSc. Thesis, Department of Operational Research, University of Southampton. Southampton, U.K.

[5]. **Popyack, J. L.** (1985), *Accelerated convergence for successive approximations in Markov decision processes*, Research report, **Drexel University**, Philadelphia.

[6]. **Ross, S. M.** (1970), *Applied probability models with optimization application*, **Holden Day**, San Francisco.

[7]. **Tijms, H. C.** (1986), *Stochastic Modelling and Analysis; A computational approach*, **Wiley**, New York.

[8]. **Van Dijk, N. M.** (1993), *Queueing networks and product forms*, **Wiley**, New York.