Operations Research Letters 49 (2021) 345–349

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

Computing the conditional entry-state distribution in Erlang loss systems

Bobby S. Nyotta *, Fernanda Bravo, M. Keith Chen

UCLA Anderson School of Management, Los Angeles, CA, United States of America

ARTICLE INFO

Article history:
Received 28 July 2020
Received in revised form 16 January 2021
Accepted 8 March 2021
Available online 18 March 2021

Keywords:
Erlang loss system
M/M/n/n queue
Birth–death Markov chain
State-dependent arrivals

ABSTRACT

We consider an Erlang loss system with state-dependent arrival rates. Given the system is in steady-state and there are j customers being served, the system operator may wish to know about the distribution of arrival states for the j customers in service. Specifically, they may want the steady-state probability that any given customer entered when the system had i servers busy, given j customers are currently being served. We term this metric the *conditional entry-state distribution* and develop an algorithm to compute it.

1. Introduction

Erlang loss systems, which are also known as finite-state, Birth–Death Markov chains or M/M/n/n queues, have a rich history in stochastic modeling and have applications to transportation, health care, and communications. We specifically consider systems where the Poisson arrival rate fluctuates with the state, or the number of busy servers. State-dependent arrivals have been used to model customer behavior related to price sensitivity and balking. This model, in its purest form, with constant, state-independent arrival and departure rates, is extensively analyzed by [6]. Brumelle [2] performs a comparable analysis, but models both state-dependent arrivals and departures, while we only consider the former. Burman [3] examines a similar Erlang loss system while relaxing the Markovian service time assumption. In any of these systems, the operator may wish to know what is the probability a customer entered when the system was empty, when the system had 1 customer, and so on, given the current state, or occupancy, of the system. We denote this metric as the "conditional entry-state distribution", where conditioning is on the system's current state. In this paper, we develop an algorithm to compute its value.

One application of where this metric can be valuable is described in [5]. The setting consists of an Erlang loss system with Poisson arrival rates that are a decreasing function of price. The author is interested in finding the optimal entry fee to charge arriving customers at each state and develops an algorithm to compute the prices that maximize the long-run average revenue per unit time. Given any feasible set of prices, the system operator may like to know *where* the revenue is coming from at each state, so the operator would like to know the expected proportion of the customers who pay the entry fee when the system is empty, has 1 customer, and so on.

In these queuing systems, many steady-state performance statistics can be computed, such as the blocking probability, the steady-state distribution, the expected number of busy servers, and more. We refer the reader to [1] for further material on computable system metrics. To the best of our knowledge, there does not exist a tractable way to compute the metric we study.

Since this work relates to the expected state at which customers enter, or arrive, to the system, the well-known "Poisson Arrivals See Time Averages" (PASTA) property [8] and the related Conditional PASTA property [7] are both relevant. The latter is pertinent since our metric also involves computing steady-state, stationary values that are conditional on the system being in a particular state. However, their work does not provide any method for computing any conditional metric, so this work compliments their result.

The paper is organized as follows. Section 2 describes the model. The algorithm and convergence proof follows in Section 3. Numerical results are detailed in Section 4, and future directions are summarized in Section 5.

2. Model

We consider an M/M/n/n queuing system with state-dependent arrival rates. This system can be modeled as a finite-state, birth-and-death Markov chain with state-space $S = \{1, \ldots,$

* Correspondence to: 110 Westwood Plaza, Gold Hall, Suite 5.01, Los Angeles, CA 90095, United States of America
E-mail addresses: bobby.nyotta.1@anderson.ucla.edu (B.S. Nyotta), fernanda.bravo@anderson.ucla.edu (F. Bravo), keith.chen@anderson.ucla.edu (M.K. Chen).

https://doi.org/10.1016/j.orl.2021.03.005
0167-6377/

$n\}$ representing the number of busy servers. We consider the system in steady-state, denote the current state as $Z \in S$, and use the indexes i, j, k to refer to arbitrary states. Customers arrive to the system according to a Poisson process with a state-dependent rate, i.e., for $Z = k$ the arrival rate is $\lambda_k > 0$. Service times for an individual server are exponentially distributed with rate μ, so we use $\mu_k = k \cdot \mu$ to denote the service rate of the entire system in state k.

We are interested in computing the steady-state probability that an in-service customer joined the system when there were i servers busy, given the system is currently in state j. Namely,

$$Pr[\text{In-service customer entered in state } i | Z = j], \ \forall i, j \in S.$$

We refer to the above probability as the *conditional entry-state probability*. To compute it, we define $\Omega_{ij} \in \mathbb{R}_+$ to be the expected number of in-service customers (out of j currently being served) who arrived when the system was in state i. To recover the conditional entry-state probability, we simply compute Ω_{ij}/j, hence hereafter we focus on obtaining Ω_{ij}. Since $\Omega_{ij} = 0$ when either $j = 0$ or $i = n$, we focus on $i \in S \setminus \{n\}$ and $j \in S \setminus \{0\}$. Thus, for each state j, we define the column vector $\Omega_j = [\Omega_{0j}, \ldots, \Omega_{n-1,j}] \in \mathbb{R}_+^n$, and we note that the entries of Ω_j sum to j: $\sum_{i=0}^{n-1} \Omega_{ij} = \Omega_j^\top \mathbf{1} = j$, where $\mathbf{1}$ is the vector of all ones. Finally, we define the matrix $\Omega = [\Omega_1, \ldots, \Omega_j, \ldots, \Omega_n] \in \mathcal{M} = \{Y \in \mathbb{R}_+^{n \times n} : \sum_i Y_{ij} = j, j = 1, \ldots, n\}$, where Y is an arbitrary matrix in the set $\mathcal{M}$, which contains matrices with positive entries where the entries of the jth column sum to j.

2.1. System dynamics

We now describe how to analytically compute Ω_{ij} by applying the conservation flow principle that is typically used to compute steady-state probabilities in Markov chains.

Before deriving the system of equations for Ω_{ij}, we first introduce some additional probabilities. The steady-state probability of a given state j can be expressed as

$$\pi_j = p_{j,j-1} \cdot \pi_{j-1} + p_{j,j+1} \cdot \pi_{j+1} \quad \forall j \in S \setminus \{0\}$$

where $p_{i,k}$ is the probability of transitioning to i from k and π_k are the steady-state probabilities. Next, we define "$j \leftarrow i$" to mean "Enter j from i". Using Bayes' rule we can then derive

$$Pr[j \leftarrow j-1 | Z = j] = \frac{p_{j,j-1} \cdot \pi_{j-1}}{p_{j,j-1} \cdot \pi_{j-1} + p_{j,j+1} \cdot \pi_{j+1}} = \frac{\mu_j}{\lambda_j + \mu_j} \tag{1}$$

The last equality is obtained by making two observations. First, transitions out of state j are with probability $p_{j-1,j} = \frac{\mu_j}{\lambda_j + \mu_j}$ to state $j-1$ if a departure occurs, or with probability $p_{j+1,j} = \frac{\lambda_j}{\lambda_j + \mu_j}$ to state $j+1$ if an arrival happens instead. Second, using the reversibility property of M/M/n/n queuing systems [4], which implies that $p_{j,j-1} = \frac{\mu_j}{\lambda_j + \mu_j} \cdot \frac{\pi_j}{\pi_{j-1}}$ and $p_{j,j+1} = \frac{\lambda_j}{\lambda_j + \mu_j} \cdot \frac{\pi_j}{\pi_{j+1}}$. For ease of exposition we hereafter define $\alpha_j := \frac{\mu_j}{\lambda_j + \mu_j}$ and $\boldsymbol{\alpha} = [\alpha_0, \ldots, \alpha_n]$, and note $\alpha_0 = 0$ and $\alpha_n = 1$.

We now proceed to write the system of equations to compute $\Omega_{i,j}$. We do this by conditioning on the two events that can lead to the system reaching state j: (1) from $j-1$ when an arrival occurs or (2) from $j+1$ when a departure occurs. Thus, the expected number of customers in state j who entered in state i ($\Omega_{i,j}$) can be expressed as a combination of: (1) the expected number of existing customers in state $j-1$ who first arrived when the system was in state i ($\Omega_{i,j-1}$) and the expected number of new customers into j who enter the system in state i ($\mathbb{I}[i = j-1]$) and (2) the expected number of existing customers in state $j+1$ who first arrived when the system was in state i ($\Omega_{i,j+1}$) minus the departing customers from state $j+1$ who first arrived when the system was in state i ($\frac{1}{j+1} \cdot \Omega_{i,j+1}$, where $\frac{1}{j+1}$ is the probability that any of the existing customers depart from the system, which is due to the memoryless property of exponential service times). Namely,

$$\begin{aligned}
\Omega_{ij} =& \left(\Omega_{i,j-1} + \mathbb{I}[i = j-1]\right) \cdot Pr[j \leftarrow j-1 | Z = j] \\
&+ \left(\Omega_{i,j+1} - \frac{1}{j+1} \cdot \Omega_{i,j+1}\right) \cdot Pr[j \leftarrow j+1 | Z = j] \\
=& (\Omega_{i,j-1} + \mathbb{I}[i = j-1]) \cdot \alpha_j \\
&+ \left(\Omega_{i,j+1} - \frac{1}{j+1} \cdot \Omega_{i,j+1}\right) \cdot (1 - \alpha_j)
\end{aligned} \tag{2}$$

We extend this flow conservation between $\Omega_{ij}, \Omega_{i,j-1}$, and $\Omega_{i,j+1}$ to matrix form in Eq. (3) with the function $g : \mathcal{M} \mapsto \mathcal{M}$. We define e_j as the unit vector with a 1 in the jth component, and use it to capture an arrival from state $j-1$ so the system evolves from having $j-1$ to j servers busy.

$$g(Y) = \begin{cases} \alpha_1(e_1) + \frac{1-\alpha_1}{2} Y_2 & \text{if } j = 1 \\ \alpha_j(Y_{j-1} + e_j) + (1 - \alpha_j)\frac{j}{j+1} Y_{j+1} & \text{if } 1 < j < n \\ Y_{n-1} + e_n & \text{if } j = n \end{cases} \tag{3}$$

To understand the intuition in g, consider the general case of $1 < j < n$: the function mixes the distribution at $j-1$ while considering an arrival into j (i.e., $Y_{j-1} + e_j$), and the distribution at $j+1$ considering the potential departures (i.e., $\frac{j}{j+1} Y_{j+1}$). We note g can be applied to any $Y \in \mathcal{M}$ and for each column Y_j, g computes a convex combination of two vectors that sum to j. This means $g(Y)$ maps to an element of $\mathcal{M}$.

3. An algorithm to compute the conditional expected entry state

Next, we present an algorithm to compute Ω. To simplify notation, we define the function $f(Y) = g(g(Y)) = Y'$ and note that $f : \mathcal{M} \mapsto \mathcal{M}$. The output $Y' = [Y'_1, \ldots, Y'_j, \ldots, Y'_n]$ has several unique, structural properties, such as its columns Y'_j are weighted, linear sum of the columns of Y, where the coefficients are polynomial functions of $\boldsymbol{\alpha}$ (Property A.1). Lemma 1 states a key property that f is a contraction mapping. For readability, all properties and proofs are available in the Appendix.

Lemma 1. *The function $f : \mathcal{M} \mapsto \mathcal{M}$ is a contraction mapping.*

We next present a simple algorithm that repeatedly applies f to any matrix in $\mathcal{M}$ until the desired tolerance ϵ is reached. Then, in Theorem 1 we show that the algorithm converges to the correct value of Ω, and in Theorem 2, we give an upper bound on the algorithm's iteration limit.

Algorithm: Computing the Conditional Expected Entry State

```
Input: ε > 0, Y ∈ M
Compute Y' = f(Y)
while d(Y, Y') = ||Y − Y'||_∞ = max_{i,j}{|Y_ij − Y'_ij|} > ε do
    Set Y = Y'
    Compute Y' = f(Y)
end
Output: Y'
```

Theorem 1. *For all $Y \in \mathcal{M}$, as $\epsilon \to 0$, the algorithm converges to Ω.*

Theorem 2. *For all $\epsilon \in (0, 1)$ and $Y = [Y_1, \ldots, Y_n] \in \mathcal{M}$, where $Y_j = \frac{j}{n} \cdot \mathbf{1}$, the algorithm terminates in at most $\log_A(\epsilon) + 1$ iterations. The parameter $A = \max_{j=1,\ldots,n}\{H_j(\boldsymbol{\alpha})\}$, where $H_j(\boldsymbol{\alpha})$, defined in Table A.1, is the maximum absolute difference in the estimate of Ω_j after one iteration.*

Theorem 2 presents an upper bound on the number of iterations the algorithm runs until convergence. However, we note that the bound is not tight. This is because A, the upper bound on the rate at which the distance d decreases in each iteration, is bounded above by $\frac{n-1}{n}$ and approaches 1 when n becomes large, so $\log_A(\epsilon)$ becomes large. When A does equal its upper bound of $\frac{n-1}{n}$, the term $\log_{\frac{n-1}{n}}(\epsilon)$ is concave increasing in n. In the next section, we observe that the numerical performance is much better.

4. Numerical experiments

In this section we numerically demonstrate the speed and efficacy of our algorithm in computing the metric of interest Ω. For several system sizes $n \in \{2, 5, 10, 25, 50, 100, 250\}$, we generate 50 random instances where $\mu \sim \mathcal{U}(0, 5)$ and $\lambda_k \sim \mathcal{U}(0, 10)$ for $k = 0, \ldots, n$. For each instance, we run the algorithm with $\epsilon = 10^{-12}$ and initial point $Y = [Y_1, \ldots, Y_k, \ldots, Y_n] \in \mathcal{M}$, where $Y_k = \mathbf{1} \cdot \frac{k}{n}$, and we record the number of iterations and time to convergence. We also simulate the system for 25,000 arrivals and compute Ω using the second half of the simulation, when the system is in steady-state. We denote this value Ω^{sim}. Using the output of the algorithm, Ω, and Ω^{sim}, we compute two metrics: the maximum and the mean absolute deviation between the Ω and Ω^{sim}. Experiments are run in Python 2.7 on a Dell Inspiron 13 with 16 GB of RAM and an Intel Core i7 1.8 GHz processor.

For each value of n, we report the average and standard deviation over the 50 random instances in Table 1. These results show the algorithm and simulation return values very close to each other. However, as n increases, the simulation rarely reaches some states in steady-state, so reliably computing Ω^{sim}_{ij} is challenging (and impossible when state j is never reached). This is evident in the last two columns of Table 1 where the deviation between Ω and Ω^{sim} increases with n because Ω^{sim}. The algorithm is never victim to this issue and always converges to the true value of Ω, which is why we can run the algorithm for all n but can only simulate for $n \leq 25$.

Table 1 also shows that the algorithm computes Ω faster than the simulation method. To capture the algorithm's convergence speed, in Fig. 1 we report the algorithm's average distance by iteration over 50 random instances. The figure shows that the algorithm converges rapidly for several values of n. We note that there is a "kink" and speed-up around the $n/2$th iteration.

One potential interpretation is related to the structure of $Y' = f(Y)$. Per Property A.1, Y'_j has two components: a weighted sum of the columns of Y and a vector C_j. With each iteration, the weights on the columns of Y approach 0, so the influence of Y effectively vanishes after many iterations. The vector C_j is a function of $\boldsymbol{\alpha}$. After one iteration, C_j is only a function of α_{j-1}, α_j, and α_{j+1}, but after two iterations, C_j is a function of $\alpha_{j-2}, \ldots, \alpha_{j+2}$. It takes at most $n/2$ iterations for C_j to be a function of all $\alpha_1, \ldots, \alpha_n$. Once C_j depends on all α_j, the vector finally captures the entire system's dynamics, so the convergence rate increases after this iteration.

5. Future work and extensions

An area for future work is to improve the bound we present for the algorithm's iteration limit. One approach worth exploring involves deriving an analytical expression to compute any value in the Cauchy sequence converging Ω_{ij}. This is different from our approach which shows a contraction modulus less than 1 exists. Two possible extensions include the cases with finite capacities (i.e. M/M/n/L queues with L > n) and with infinite capacity (i.e. $L = \infty$).

Table 1
Comparison of algorithm and simulation.

n	Iterations	Algorithm time in seconds	Simulation time in seconds	$\max_{ij} \lvert\Omega_{ij} - \Omega^{sim}_{ij}\rvert$	$\frac{1}{n^2}\sum_{ij} \lvert\Omega_{ij} - \Omega^{sim}_{ij}\rvert$
2	24.66 (4.03)	0.26 (0.05)	50.42 (1.24)	0.009 (0.006)	0.007 (0.005)
5	57.34 (14.28)	1.00 (0.25)	67.19 (2.37)	0.084 (0.039)	0.020 (0.006)
10	68.78 (22.22)	2.02 (0.65)	92.43 (4.04)	0.914 (0.395)	0.100 (0.034)
25	84.44 (27.2)	5.45 (1.76)	159.92 (7.86)	1.247 (0.505)	0.275 (0.014)
50	99.00 (31.74)	12.22 (3.92)	– –	– –	– –
250	200.66 (26.57)	125.79 (16.58)	– –	– –	– –

Standard deviation values are shown in parenthesis.

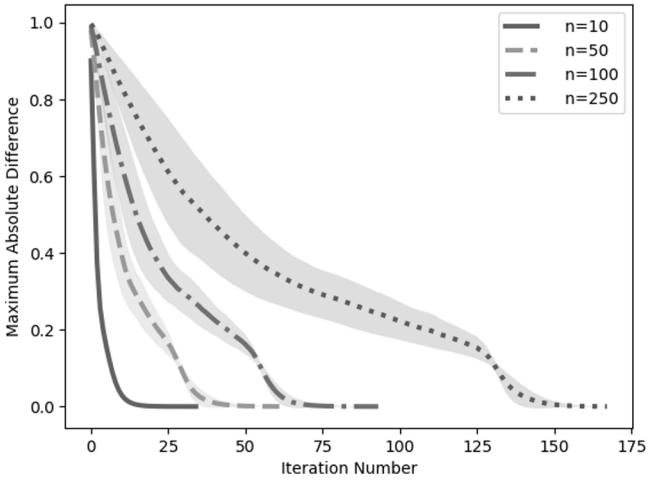

Fig. 1. Mean convergence rate with shaded standard deviation for various n. *Note*. We report average distance values for all iterations up until the quickest random instance terminates.

Acknowledgments

The authors thank The Easton Technology Management Center and The Price Center for Entrepreneurship and Innovation at the UCLA Anderson School of Management for supporting this research.

Appendix. Properties of f and proofs

Property A.1. *Let $X \in \mathcal{M}$ and $X' = f(X)$.*

1. *(Affine Structure) Each column in X' has the form: $X'_j = C_j + \sum_{l=1}^{n} b_{j,l} \cdot X_l$, where $C_j \in \mathbb{R}^n_+$ and $b_{j,l} \in \mathbb{R}_+$ are polynomial functions of $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]$.*

Proof of Lemma 1. For all $X, Y \in \mathcal{M}$, we first define the distance metric $\Delta = d(X, Y) = \max_{i,j}\{|X_{ij} - Y_{ij}|\} = \|X - Y\|_\infty$. We note that $(\mathcal{M}, d)$ is a non-empty complete metric space. Equivalently, $\Delta = \max_j\{\Delta_j\}$ where $\Delta_j = d(X_j, Y_j)$. Let $X' = f(X)$, $Y' = f(Y)$, and $\Delta' = d(X', Y')$. To show that f is a contraction on $\mathcal{M}$, we show that for all $X, Y \in \mathcal{M}, \exists\, 0 \leq q < 1$ s.t. $\Delta' \leq q \cdot \Delta$. Since $\Delta' = \max_j\{\Delta'_j\}$, we will show that for all $j, \exists\, 0 \leq q < 1$ s.t. $\Delta'_j \leq q \cdot \Delta$.

Table A.1
Expressions for X'_j and $H_j(\boldsymbol{\alpha})$.

j	X'_j	$H_j(\boldsymbol{\alpha})$
1	$\alpha_1 \cdot e_1 + \frac{1}{2}(1-\alpha_1)\alpha_2 \cdot e_2 + \frac{1}{2}(1-\alpha_1)\alpha_2 \cdot X_1 + \frac{1}{3}(1-\alpha_1)(1-\alpha_2) \cdot X_3$	$\frac{1}{2}(1-\alpha_1)\alpha_2 + \frac{1}{3}(1-\alpha_1)(1-\alpha_2)$
2	$\alpha_1\alpha_2 \cdot e_1 + \frac{2}{3}(1-\alpha_2)\alpha_3 \cdot e_3 + \left(\frac{1}{2}(1-\alpha_1)\alpha_2 + \frac{2}{3}(1-\alpha_2)\alpha_3\right) \cdot X_2 + \frac{1}{2}(1-\alpha_2)(1-\alpha_3) \cdot X_4$	$\frac{1}{2}(1-\alpha_1)\alpha_2 + \frac{2}{3}(1-\alpha_2)\alpha_3 + \frac{1}{2}(1-\alpha_2)(1-\alpha_3)$
$2 < j < n-1$	$\alpha_j\alpha_{j-1} \cdot e_{j-1} + \left(\frac{j}{j+1}(1-\alpha_j)\alpha_{j+1} + \alpha_j\right) \cdot e_j + \alpha_j\alpha_{j-1} \cdot X_{j-2} + \left(\frac{j-1}{j}(1-\alpha_{j-1})\alpha_j + \frac{j}{j+1}(1-\alpha_j)\alpha_{j+1}\right) \cdot X_j + \frac{j}{j+2}(1-\alpha_j)(1-\alpha_{j+1}) \cdot X_{j+2}$	$\alpha_j\alpha_{j-1} + \frac{j-1}{j}(1-\alpha_{j-1})\alpha_j + \frac{j}{j+1}(1-\alpha_j)\alpha_{j+1} + \frac{j}{j+2}(1-\alpha_j)(1-\alpha_{j+1})$
$n-1$	$\alpha_{n-2}\alpha_{n-1} \cdot e_{n-2} + \alpha_{n-1} \cdot e_{n-1} + \frac{n-1}{n}(1-\alpha_{n-1}) \cdot e_n + \alpha_{n-2}\alpha_{n-1} \cdot X_{n-3} + \left(\frac{n-2}{n-1}(1-\alpha_{n-2})\alpha_{n-1} + \frac{n-1}{n}(1-\alpha_{n-1})\right) \cdot X_{n-1}$	$\alpha_{n-2}\alpha_{n-1} + \frac{n-2}{n-1}(1-\alpha_{n-2})\alpha_{n-1} + \frac{n-1}{n}(1-\alpha_{n-1})$
n	$\alpha_{n-1} \cdot e_{n-1} + e_n + \alpha_{n-1} \cdot X_{n-2} + \frac{n-1}{n}(1-\alpha_{n-1}) \cdot X_n$	$\alpha_{n-1} + \frac{n-1}{n}(1-\alpha_{n-1})$

To illustrate how we derive X'_j, consider the case $j = 1$: Let $g(X)_j$ be the value of the jth column after one application of g. With this, $X'_1 = f(X)_1 = g(g(X))_j$. By Eq. (3), $X'_1 = \alpha_1(e_1) + \frac{1-\alpha_1}{2} g(X)_2$, and $g(X)_2 = \alpha_2(X_1 + e_2) + (1-\alpha_2)\frac{2}{3}X_3$. Combining this, we get $X'_1 = \alpha_1(e_1) + \frac{1-\alpha_1}{2}(\alpha_2(X_1 + e_2) + (1-\alpha_2)\frac{2}{3}X_3)$, the expression above.

The proof has four steps: (1) we upper bound Δ'_j by a constant times Δ. (2) we show this constant is a function of $\boldsymbol{\alpha}$. In (3) and (4), we show this constant is always strictly less than 1, proving f is a contraction.

(1) Upperbound on Δ'_j. For an arbitrary j, we can find an upperbound on Δ'_j as follows:

$$\begin{aligned}
\Delta'_j = \max_i |X'_{ij} - Y'_{ij}| &= \max_i |C_{ij} + \sum_{l=1}^n b_{j,l} \cdot X_{il} - C_{ij} - \sum_{l=1}^n b_{j,l} \cdot Y_{il}| \\
&= \max_i |\sum_{l=1}^n b_{j,l} \cdot (X_{il} - Y_{il})| \\
&\le \max_i \sum_{l=1}^n |b_{j,l}| \cdot |X_{il} - Y_{il}| \\
&\le \sum_{l=1}^n |b_{j,l}| \cdot \max_i |X_{il} - Y_{il}| \\
&= \sum_{l=1}^n b_{j,l} \cdot \Delta_l \le \sum_{l=1}^n b_{j,l} \cdot \Delta
\end{aligned}$$

The second and last equality in the above equation come from Property A.1. This shows that $\Delta'_j \le \sum_{l=1}^n b_{j,l} \cdot \Delta$, where $\sum_{l=1}^n b_{j,l} \in \mathbb{R}_+$. If $\sum_{l=1}^n b_{j,l} < 1$ for all j, then f is a contraction.

(2) Expressing $\sum_{l=1}^n b_{j,l}$ **as a Function of** $\boldsymbol{\alpha}$. For the general case when $n \ge 5$, the columns in the output matrix $X' = f(X)$ for an arbitrary $X \in \mathcal{M}$ fall into one of five categories based on the column number j: left columns ($j = 1, 2$), right columns ($j = n-1, n$), and interior columns ($2 < j < n-1$). Using Eq. (3) and f, we derive the expressions for X'_j in terms of $\boldsymbol{\alpha}$ and present them in Table A.1. We also define the function $H_j = \sum_{l=1}^n b_{j,l}$ as a function of $\boldsymbol{\alpha}$. We note when $n = 2$, we use columns X'_1 and X'_n. When $n = 3$, we use columns X'_1, X'_{n-1}, X'_n. For $n = 4$, we use columns $X'_1, X'_2, X'_{n-1}, X'_n$.

(3) Showing Maximum Value of H_j **is Strictly Less Than 1.** For the five cases described in Table A.1, we want to show $\max_{\boldsymbol{\alpha} \in [0,1]^n : 0 < \alpha_k < 1} \{H_j(\boldsymbol{\alpha})\} < 1$. We refer to this as Problem A and denote z_A its optimal value and $\boldsymbol{\alpha}^*_A$ its maximizer. However, problem A's feasible region is not a compact set due to the strict inequalities on $\boldsymbol{\alpha}$. Relaxing the strict inequalities in A, we have $\max_{\boldsymbol{\alpha} \in [0,1]^n : 0 \le \alpha_k \le 1} \{H_j(\boldsymbol{\alpha})\}$, which we refer to problem B. We define z_B and $\boldsymbol{\alpha}^*_B$ similarly. Since A's feasible region is strictly contained within B's, we have $z_A \le z_B$, and since B's feasible region is compact, we can solve for z_B and $\boldsymbol{\alpha}^*_B$. If we show that $z_B < 1$ or $z_A < z_B = 1$ for all j, then we have shown f is a contraction. Below, we compute z_B and α^*_B for each case.

$j = 1$: $z_B = \max_{0 \le \alpha_1, \alpha_2 \le 1} \{H_1(\alpha_1, \alpha_2)\} = \frac{1}{2}$ since $\frac{\partial H_1}{\partial \alpha_2} = \frac{1-\alpha_1}{6} \ge 0$, so $\alpha^*_B = (0, 1)$ and $z_B = \frac{1}{2}$.

$j = 2$: $z_B = \max_{0 \le \alpha_1, \alpha_2, \alpha_3 \le 1} \{H_2(\alpha_1, \alpha_2, \alpha_3)\} = \frac{2}{3}$ since $\frac{\partial H'_2}{\partial \alpha_1} = \frac{-\alpha_2}{2} \le 0$, $\frac{\partial H'_2}{\partial \alpha_2} = \frac{-\alpha_1}{2} - \frac{\alpha_3}{6} \le 0$, and $\frac{\partial H'_2}{\partial \alpha_3} = \frac{1-\alpha_2}{6} \ge 0$. Therefore, $\alpha^*_B = (0, 0, 1)$ and $z_B = \frac{2}{3}$.

$2 < j < n-1$: $z_B = \max_{0 \le \alpha_{j-1}, \alpha_j, \alpha_{j+1} \le 1} \{H_j(\alpha_{j-1}, \alpha_j, \alpha_{j+1})\} = 1$ since $\frac{\partial H_j}{\partial \alpha_{j-1}} = \frac{\alpha_j}{j} \ge 0$ and $\frac{\partial H_j}{\partial \alpha_{j+1}} = \frac{j(1-\alpha_j)}{(j+1)(j+2)} \ge 0$, so $\alpha^*_{j-1,B} = \alpha^*_{j+1,B} = 1$. Then $\frac{\partial H_j}{\partial \alpha_j} = \frac{j-2}{j(j+2)} + \frac{\alpha_{j-1}}{j} - \frac{\alpha_{j+1} \cdot j}{(j+2)(j+1)} > 0$, so $\alpha^*_B = (1, 1, 1)$ and $z_B = 1$.

$j = n-1$: $z_B = \max_{0 \le \alpha_{n-2}, \alpha_{n-1} \le 1} \{H_{n-1}(\alpha_{n-2}, \alpha_{n-1})\} = 1$ since $\frac{\partial H_{n-1}}{\partial \alpha_{n-2}} = \frac{1}{n}\alpha_{n-1} \ge 0$, which implies $\alpha^*_{n-2,B} = 1$. When $\alpha_{n-2} > \frac{1}{n}$, we have $\frac{\partial H_{n-1}}{\partial \alpha_{n-1}} = \frac{\alpha_{n-2}}{n-1} - \frac{1}{n(n-1)} > 0$. Therefore, $\alpha^* = (1, 1)$ and $z_B = 1$.

$j = n$: $z_B = \max_{0 \le \alpha_{n-1} \le 1} \{H_n(\alpha_{n-1})\} = 1$ since $\frac{\partial H_n(\alpha_{n-1})}{\partial \alpha_{n-1}} = \frac{1}{n} \ge 0$, so $\alpha^*_{n-1,B} = 1$ and $z_B = 1$.

(4) Final Step. For $j = 1, 2$, $z_B < 1$, so we are done for those cases. However, for the cases $j > 2$, $z_B = 1$. All of the solutions in these cases are corner solutions where the elements of $\boldsymbol{\alpha}^*_B$ take either 0 or 1. Since problem A's feasible region is strictly contained in B's, the corner solution $\boldsymbol{\alpha}^*_B$ cannot be reached in A. Therefore, $z_A < z_B = 1$ for all $j \in \{1, \ldots, n\}$ and f is a contraction. □

Proof of Theorem 1. From Lemma 1 and the Banach fixed-point theorem, we know there exists a unique $Y^* \in \mathcal{M}$ such that $Y^* = f(Y^*)$ and $d(Y^*, f(Y^*)) = \|Y^* - f(Y^*)\|_\infty = 0$. We also know $d(Y, Y') \ge d(Y', Y'') \ge \cdots > 0$, where $Y^{(k)} = f^{\circ k}(X)$ for $k \in \mathbb{Z}_+$. For each application of f, or as $k \to \infty$, the distance d successively decreases. Therefore as $\epsilon \to 0$, the algorithm converges to Y^*.

Next, we show $Y^* = \Omega$. Assume that $Y^* \ne \Omega$. Embedded within f is g, which uses steady-state transition probabilities to model the flow between states, so we know $g(\Omega) = \Omega$, which means that $f(\Omega) = g(g(\Omega)) = \Omega$. If $Y^* \ne \Omega$, then $f(Y^*) \ne Y^*$ and $d(Y^*, f(Y^*)) > 0$, which is impossible since Y^* is the fixed point of the contraction mapping f. Therefore $Y^* = \Omega$ and the algorithm converges to the correct Ω as $\epsilon \to 0$. □

Proof of Theorem 2. If d_1 is the distance after the first iteration of the algorithm and q is the number of subsequent iterations until the distance reaches the stopping tolerance ϵ, then the algorithm terminates when $d_1 \cdot A^q < \epsilon$, where $A = \max_{j=1,\ldots,n} \{H_j(\boldsymbol{\alpha})\} \in (0, 1)$. Note that $H_j(\boldsymbol{\alpha})$ (see Table A.1) is the upper bound on the rate at which the distance decreases in successive iterations.

Rearranging the expression, we have $q < \log_A(\epsilon/d_1)$, so the algorithm terminates in, at most, $\log_A(\epsilon/d_1)+1$ iterations. When the input matrix is $Y = [Y_1, \ldots, Y_j, \ldots, Y_n] \in \mathcal{M}$, where $Y_j = \frac{j}{n}\cdot\mathbf{1}$, then the distance $d_1 = d(Y, Y') \to 1$ as $n \to \infty$. Applying this insight, we know the algorithm terminates in at most $\log_A(\epsilon)+1$ iterations. □

References

[1] A.O. Allen, Probability, Statistics, and Queueing Theory, Academic Press, 2014.
[2] S.L. Brumelle, A generalization of erlang's loss system to state dependent arrival and service rates, Math. Oper. Res. 3 (1) (1978) 10–16.
[3] D.Y. Burman, Insensitivity in queueing systems, Adv. Appl. Probab. (1981) 846–859.
[4] F.P. Kelly, Reversibility and Stochastic Networks, Cambridge University Press, 2011.
[5] D.W. Low, Optimal dynamic pricing policies for an m/m/s queue, Oper. Res. 22 (3) (1974) 545–561.
[6] L. Takacs, On erlang's formula, Ann. Math. Stat. 40 (1) (1969) 71–78.
[7] E.A. Van Doorn, G. Regterschot, Conditional PASTA, Oper. Res. Lett. 7 (5) (1988) 229–232.
[8] R.W. Wolff, Poisson arrivals see time averages, Oper. Res. 30 (2) (1982) 223–231.