Modelling Queues Where Customers Randomly Change Priority Classes While Waiting

Geraint I. Palmer, Michalis Panayidis & Vincent Knight

1 Introduction

There are a number of situations in which a customer's priority in a queue might change during their time queueing, or equivalently where their priority depends on the amount of time they have already spent in the queue. Classic examples arise in healthcare systems, for example when a patient's medical urgency might increase the longer they spend waiting due to health degeneration. Another example would be a prioritisation scheme that attempts a trade-off between medical need and waiting times. These are both examples where a patient's priority has the chance to upgrade over time while in the queue. However there also might be situations in which a patient's priority can downgrade over time: consider a medical intervention that can improve a patient's outcome if caught early, if a patient has been waiting a long time already then they might be passed over for a newly referred patient who will gain more benefit from the intervention. In this case a patient's priority is downgraded the longer they wait.

In this paper a single $M/M/c$ queue is modelled, with multiple classes of customer of different priorities. While waiting in the queue customers change their class to any other class at specific rates. Thus upgrades, downgrades, and 'skip-grades' (moving to a priority class not immediately above or below the current class) are modelled.

This is first modelled using simulation, where we describe generalisable logic. This is implemented in version v2.3.0 of the Ciw library in Python [14]. Then two Markov chain models are defined for the system, which are used to find steady state distributions and expected sojourn times for each customer class. These Markov chains give some insights into the behaviour of the systems under different combinations of parameters; and numerical experiments give further behaviours.

This paper is structured as follows: Section 2 defines the system under consideration in detail. Section 3 highlights some previous and related work. Section 4 discusses the contribution to the Ciw library and the logic required to simulate this system. Section 5 defines two Markov chain models of the system, one useful for considering system-wide statistics such as state probabilities, and one useful for considering customers' statistics such as average sojourn time. Section 6 explores a bounded approximation for numerically tractable analysis, and gives guidelines on choosing a large enough bound so as to sufficiently approximate an unbounded system. Section 7 presents results from numerical experiments that give insight into the behaviour of the system under various parameters. Section 8 uses these models to model scenarios where prioritisation rules are unknown.

2 System Under Consideration

Consider an $M/M/c$ queue with K classes of customer labelled $0, 1, 2, \dots, K-1$. Let:

- λ_k be the arrival rate of customers of class k,
- μ_k be the service rate of customers of class k,

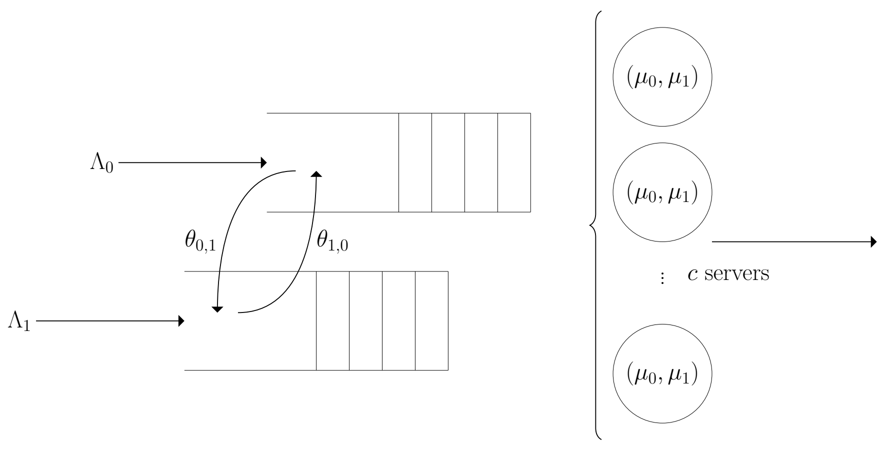


Figure 1: An example of a two-class priority queue.

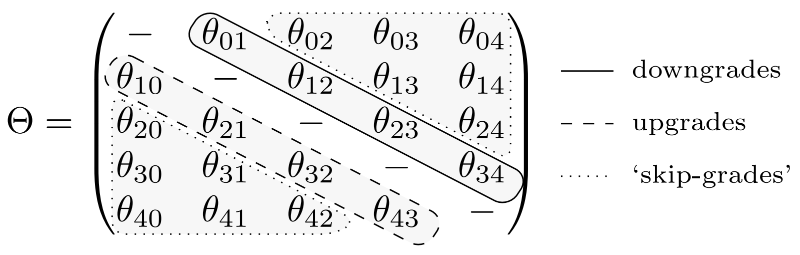


Figure 2: Representations of parts of the matrix Θ. Example when $K = 5$.

- $\theta_{i,j}$ be the rate at which customers of class i change to customers of class j while they are waiting in line.

Customers of class i have priority over customers of class j if $i < j$. Customers of the same class are served in the order they arrived to that class.

Figure 1 shows an example with two classes of customer.

The key feature here is the $K \times K$ class change matrix $\Theta = (\theta_{i,j})$. All elements $\theta_{i,j}$ where $i \neq j$ are rates, and so are non-negative real numbers, if customers of class i cannot change to customers of class j directly, then $\theta_{i,j} = 0$. The diagonal values $\theta_{i,i}$ are unused as customers cannot change to their own class. All elements $\theta_{i,i-1}$ represent the direct upgrade rates; all elements $\theta_{i,i+1}$ represent the direct downgrade rates, while all other elements can be thought of as 'skip-grades'. This is shown in Figure 2.

3 Related Work

Systems of this kind have been investigated previously:

- [7] (1960): Non preemptive M/M/1 where customers are served in order of the difference between their waiting time and urgency number (that is priorities increasing linearly over time). Solved by considering event probabilities at clock ticks.
- [6] (1971): Similar to the above, but treat each urgency number as a separate customer class, and not considering clock ticks. Upper and lower bounds on the waiting times, based on FIFO and static priorities.

- [12] (1979): Now considers the case where priorities increase non-linearly but concavely over time.
- [4] (1990): Non preemptive M/G/1 queue with two classes of customers, where priorities switch if the number from one class exceeds a given threshold. Lower priority customers have a finite waiting capacity, higher have infinite capacity.
- [9] (2003): Similar to the above but with Markovian services and infinite waiting capacities for both customers.
- [16] (2008): Preemptive n-priority-classes M/M/c with exponential upgrades. Customers only upgrade to the priority immediately higher than themselves. Stability considered.
- [3] (2010): Preemptive two-priority-classes M/M/c with exponential upgrades. Customers cannot upgrade if the number of lower priority customers is below a given threshold. Holding costs considered.
- [5] (2012): Extension of the above, allows batch arrivals, multiple classes, phase-type upgrades and services. Customers only upgrade to the priority immediately higher than themselves.
- [1] (2022): Analytical (truncated) expressions for a two class delayed accumulating priority M/G/1 queue. Customer priorities increase linearly over time, at different rates according to class, after an initial fixed delay.

Add more literature, flesh out as paragraphs.

4 Simulation Model Logic

The Ciw library [14] is an open-source Python library for discrete-event simulation of open queueing networks. A key contribution of this work is the adaptation of the library's logic to facilitate the type of dynamically changing priority classes described in Section 2. This adaptation is released in version Ciw v2.3.0, with usage documentation at `https://ciw.readthedocs.io/en/latest/Guides/change-class-while-queueing.html`.

The core of Ciw's logic is the event scheduling approach, described in [15]. This is a variant of the three-phase approach, with an **A**-phase which advances the clock to the next scheduled event, a **B**-phase where scheduled events are carried out, and a **C**-phase where conditional events are carried out. Figure 3 shows a flow diagram of the logic of the event scheduling approach.

The primary scheduled, or **B**-events are customers arriving to a queue, and customers finishing service. The conditional, or **C**-events are those that happen immediately after, and because of, these **B**-events. The primary ones are customers beginning service, and customers leaving the queue.

All other features of queueing systems that can be simulated with the Ciw library involve increasing the range of **B**- and **C**-events that can happen during the simulation run. In the case of customers randomly changing priority classes while waiting, one additional **B**-event and one additional **C**-event are included:

- Upon arrival to the queue customers are assigned a date in which they will change customer class, determined by randomly sampling from a distribution. Therefore each customer's event of changing customer class is scheduled for the future, and are therefore **B**-events. If those customers begin service (which might not be scheduled yet) before that event has occurred, then their changing customer class event is cancelled.
- Upon changing class, they immediately schedule another changing class event for the future, again sampling a date from a given distribution. This happens immediately after the above, and so is a **C**-event.

Note that the particular distributions used to sample class change dates in these cases are generic, and any of Ciw's currently pre-programmed distributions can be chosen, or custom distributions can also be input.

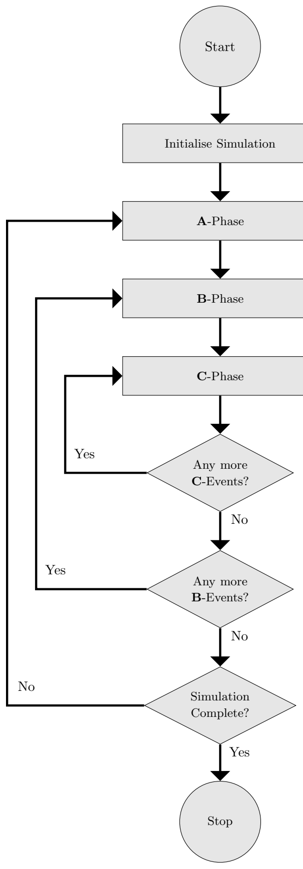


Figure 3: Flow diagram of the event scheduling approach used by Ciw, taken from [13].

For the systems described in this paper, we choose Exponential distributions with rates determined by the class change matrix Θ.

5 Markov Chain Models

The situation described in words in Section 2 can be described precisely as two different Markov chains. The first, described in Section 5.1, describes the overall changes in state, where a state records the number of customers of each class at the node. This is useful for analysing system-wide statistics such as average queue size. The second, described in Section 5.2, describes how an individual arriving customer experiences the system until their exit. This is useful for analysing individual customers' statistics such as average sojourn time.

5.1 Discrete State Markov Chain Formulation

Let $\underline{\mathbf{s}}_t = (s_{0,t}, s_{1,t}, \dots, s_{K-1,t}) \in \mathbb{N}^K$ represent the state of the system at time step t, where $s_{k,t}$ represents the number of customers of class k present at time step t.

Then the rates of change between $\underline{\mathbf{s}}_t$ and $\underline{\mathbf{s}}_{t+1}$ are given by Equation 1, where $\underline{\delta} = \underline{\mathbf{s}}_{t+1} - \underline{\mathbf{s}}_t$,

$$q_{\underline{\mathbf{s}}_t, \underline{\mathbf{s}}_{t+1}} = \begin{cases} \lambda_k & \text{if } \delta_k = 1 \text{ and } \delta_i = 0 \; \forall \; i \neq k \\ B_{k,t}\mu_k & \text{if } \delta_k = 1 \text{ and } \delta_i = 0 \; \forall \; i \neq k \text{ and } \sum_{i<k} s_{i,t} < c \\ (s_{k,t} - B_{k,t})\theta_{k_0,k_1} & \text{if } \delta_{k_0} = -1 \text{ and } \delta_{k_1} = 1 \text{ and } \delta_i = 0 \; \forall \; i \neq k_0, k_1 \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

and $B_{k,t}$, representing the number of customers of class k currently in service at time step t, is given by Equation 2, where c is the number of servers.

$$B_{k,t} = \min\left(c - \min\left(\sum_{i<k} s_{i,t}, c\right), s_{k,t}\right) \tag{2}$$

5.2 Sojourn Time Markov Chain Formulation

Let $\underline{\mathbf{z}}_t = (z_{0,t}, z_{1,t}, \dots, z_{n,t} \dots, z_{K-1,t}, m_t, n_t) \in \mathbb{N}^{K+2}$ represent the state of a particular customer at time step t, where n_t represents that customer's class at time t; $z_{k,t} \; \forall \; k < n$ represents the number of customers of class k in front of the customer in the queue at time t; $z_{k,t} \; \forall \; n < k < K$ represents the number of customers of class k behind the customer in the queue at time t; and m_t represent the number of customers of class n_t behind the customer in the queue at time t. Also let $\star$ represent an absorbing state, representing the state where that customer has finished service and left the system.

Then the rates of change between $\underline{\mathbf{z}}_t$ and $\underline{\mathbf{z}}_{t+1}$ are given by Equation 3, where $\underline{\delta} = \underline{\mathbf{z}}_{t+1} - \underline{\mathbf{z}}_t$,

$$q_{\underline{\mathbf{z}}_t, \underline{\mathbf{z}}_{t+1}} = \begin{cases} \mu_n & \text{if } z_{t+1} = \star \text{ and } \sum_{k \leq n} z_{k,t} < c \\ \lambda_n & \text{if } \delta_K = 1 \text{ and } \delta_i = 0 \; \forall \; i \neq K \\ \lambda_k & \text{if } \delta_k = 1 \text{ and } \delta_i = 0 \; \forall \; i \neq k \text{ and } k \neq n \\ A_{k,n,t}\mu_k & \text{if } \delta_k = -1 \text{ and } \delta_i = 0 \; \forall \; i \neq k \text{ and } k < K \\ \tilde{A}_{n,t}\mu_n & \text{if } \delta_K = -1 \text{ and } \delta_i = 0 \; \forall \; i \neq K \\ (z_{k_0,t} - A_{k_0,n,t})\theta_{k_0,k_1} & \text{if } \delta_{k_0} = -1 \text{ and } \delta_{k_1} = 1 \text{ and } \delta_i = 0 \; \forall \; i \neq k_0, k_1 \text{ and } k_0 < K \text{ and } k_1 \neq n, K, K+1 \\ (z_{K,t} - \tilde{A}_{n,t})\theta_{n,k} & \text{if } \delta_K = -1 \text{ and } \delta_k = 1 \text{ and } \delta_i = 0 \; \forall \; i \neq k, n \text{ and } k < K \\ (z_{k,t} - A_{k,n,t})\theta_{k,n} & \text{if } \delta_k = -1 \text{ and } \delta_K = 1 \text{ and } \delta_i = 0 \; \forall \; i \neq k, K \\ \theta_{n,k} & \text{if } \delta_n = z_{K,t} \text{ and } \delta_K = -z_{K,t} \text{ and } \delta_{K+1} = n - k \text{ and } \delta_i = 0 \text{ otherwise, and } \sum_{k \leq n} z_{k,t} < c \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

and $A_{k,n,t}$ and $\tilde{A}_{n,t}$, representing the number of customers of class k currently in service, are given by Equations 4 and 5.

$$A_{k,n,t} = \begin{cases} \min\left(c, \sum_{i \leq k} z_{i,t}\right) - \min\left(c \sum_{i<k} z_{i,t}\right) & \text{if } k \leq n \\ \min\left(c, \sum_{i \leq k} z_{i,t} + 1 + z_{K,t}\right) - \min\left(c \sum_{i<k} z_{i,t} + 1 + z_{K,t}\right) & \text{if } n < k < K \end{cases} \tag{4}$$

$$\tilde{A}_{n,t} = \min\left(c, \sum_{i \leq n} z_{i,t} + 1 + z_{K,t}\right) - \min\left(c, \sum_{i \leq n} z_{i,t} + 1\right) \tag{5}$$

The expected time to absorption can be calculated from each state. Customers arrive in all states where $z_{K,t} = 0$, and their class can be determined by n. Combining these times to absorption with the state probabilities found in the previous section, the sojourn times for each customer class can be found.

6 Bounded Approximation

In order to analyse the above Markov chain models numerically, finite approximations can be made. Define b as a bound, such that the maximum allowed number of customers of each priority class is b, with transitions to any states greater than this disallowed. Assuming the infinite system reaches a stationary distribution, then the probability of the number of customers of a particular customer class exceeding b approaches zero as b increases. Thus the finite approximations approaches a good approximation as b increases, and any statistics found using a bounded approximation will approach their corresponding values of the infinite models as b increases.

Reword this and justify better.

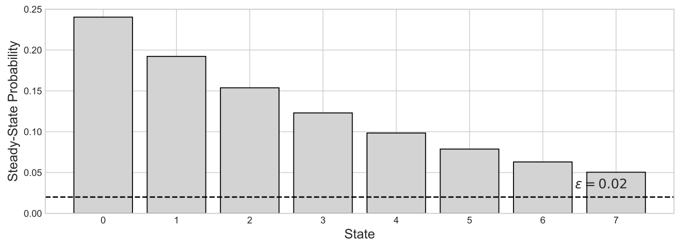
Choosing an appropriate value for b is a trade off between accuracy and model size, and so computational time. An inefficient way to choose b would be to sequentially build bounded models, increasing b each time, calculating the statistics of interest, and observing when the relationship between b and that statistic levels off. It would be more efficient to choose a b and be able to immediately check if the accuracy is sufficient. We propose two checks, one for ergodic Markov chains (the model in Section 5.1), and one for absorbing Markov chains (the model in Section 5.2).

6.1 Bound check for ergodic Markov chains

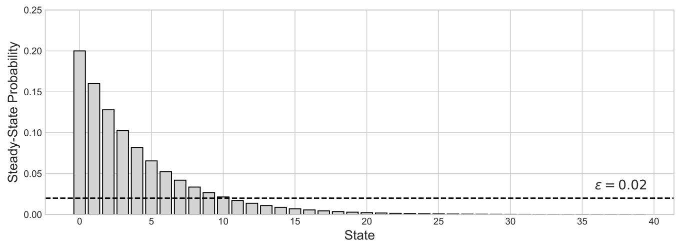
Let $S_b = \{\underline{\mathbf{s}} \in S \mid b \in \underline{\mathbf{s}}\}$, the set of states that lie on the Markov chain boundary. We wish to choose b large enough that the boundary is irrelevant, that is that the Markov chain hardly ever reaches the boundary. Consider $\max_{\underline{\mathbf{s}} \in S_b} \pi_{\underline{\mathbf{s}}}$, the largest of the steady-state probabilities of being at a state on the boundary. Choose a tolerance level $\epsilon > 0$. Now consider b large enough if

$$\max_{\underline{\mathbf{s}} \in S_b} \pi_{\underline{\mathbf{s}}} < \epsilon. \tag{6}$$

As an example, consider a standard M/M/1 queue with arrival rate $\lambda = 4$ and service rate $\mu = 5$. Here the set S_b contains just one state, the state b. Figure 4a shows the steady state probabilities obtained when $b = 7$, while Figure 4b shows the steady state probabilities obtained when $b = 39$; both choosing $\epsilon = 0.02$. Under the proposed check it can be seen that $b = 7$ is not sufficient to approximate an infinite system as $\pi_7 > \epsilon$, while $b = 39$ is sufficient as $\pi_{39} < \epsilon$.



(a) $b = 7$



(b) $b = 39$

Figure 4: Steady-state probabilities of bounded approximations of an M/M/1 queue.

Add a plot of max $\max \pi_s - \epsilon$ as a function of b.

6.2 Bound check for absorbing Markov chains

The above check isn't possible for absorbing Markov chains as they will not reach steady state, so another check is required. This method assumed the absorbing Markov chain takes the same shape as defined in Section 5.2, where states move further away from the absorbing state as their labels increase.

Again let $S_b = \{\underline{\mathbf{s}} \mid b \in \underline{\mathbf{s}}\}$, the set of states that lie on the Markov chain boundary. Now define $h_{i,J}$ as the hitting probabilities of a set of states J from state i, that is, what is the probability of ever reaching any

$$\begin{tikzcd}
\mathbf{(0)} & \mathbf{(1)} \arrow[l, bend left, "\mu"] \arrow[r, bend left, "\lambda"] & \mathbf{(2)} \arrow[l, bend left, "\mu"] \arrow[r, bend left, "\lambda"] & \mathbf{(3)} \arrow[l, bend left, "\mu"] \arrow[r, bend left, "\lambda"] & \cdots \arrow[l, bend left, "\mu"] \arrow[r, bend left, "\lambda"] & \boldsymbol{(b-1)} \arrow[l, bend left, "\mu"] \arrow[r, bend left, "\lambda"] & \boldsymbol{(b)} \arrow[l, bend left, "\mu"]
\end{tikzcd}$$

Figure 5: An M/M/1 queue with absorbing state.

state in J when starting from state i. These are defined recursively as:

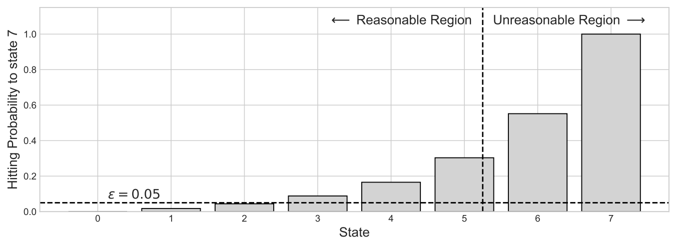
$$h_{iJ} = \begin{cases} \sum_k p_{ik} h_{kJ} & \text{if } i \notin J \\ 1 & \text{if } i \in J \end{cases} \tag{7}$$

Now for the bounded approximation define a *reasonable region*, R, of the state space, a region of states which we reasonable expect the infinite Markov chain to visit. Let it's complement be known as the *unreasonable region*. Let us define R by the percentage r of the distance between the absorbing state and a boundary state, that is $R = \{\underline{\mathbf{s}} \mid \|\mathbf{s}\| < rb \text{ for all } s \in \underline{\mathbf{s}}\}$, where $\|\mathbf{x}\| = \sum_i x_i$.

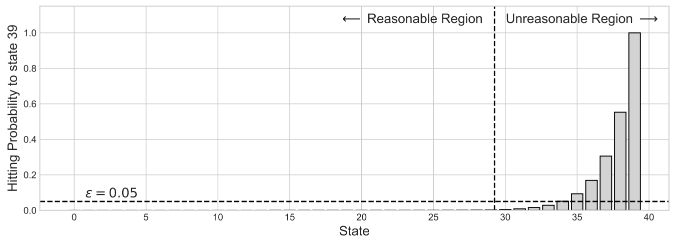
Now as above, choose a tolerance level $\epsilon > 0$. Now consider b large enough if

$$\max_{\underline{\mathbf{s}} \in R} h_{\underline{\mathbf{s}} S_b} < \epsilon. \tag{8}$$

As an example, consider a standard M/M/1 queue, with arrival rate $\lambda = 4$ and service rate $\mu = 5$, but with arrivals rejected if the queue is empty. That is the state 0 is an absorbing state, see Figure 5. Here the set S_b contains just one state, the state b. Figure 6a shows the hitting probabilities from each state to b obtained when $b = 7$, while Figure 6b shows the hitting probabilities obtained when $b = 39$; both choosing $\epsilon = 0.05$ and $r = 0.75$. Under the proposed check it can be seen that $b = 7$ is not sufficient to approximate an infinite system as $h_{5,7} > \epsilon$, while $b = 39$ is sufficient as $h_{29,39} < \epsilon$.



(a) $b = 7$



(b) $b = 39$

Figure 6: Hitting probabilities of bounded approximations of an absorbing M/M/1 queue.

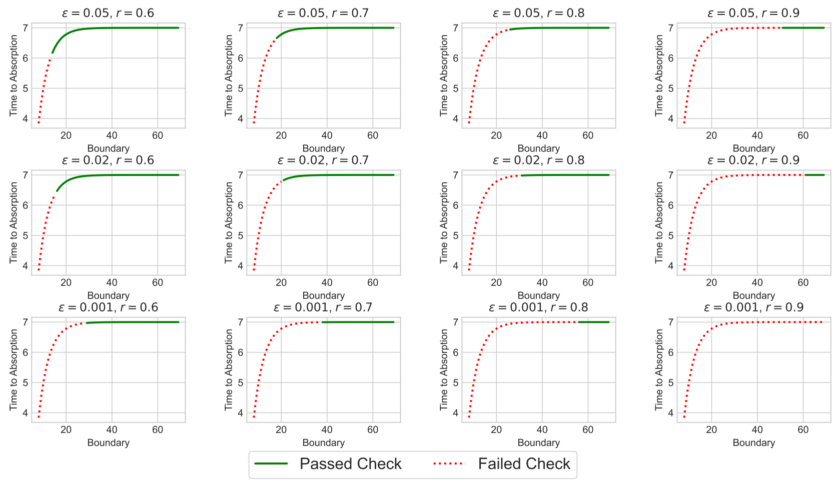


Figure 7: Investigating the effect of the hyperparameters ϵ and r.

Add a plot of max max $h - \epsilon$ as a function of b.

The hyperparameters choices of ϵ and r can greatly effect the effectiveness of this check, and aren't as intuitive as the boundary hitting tolerance hyperparameter of Section 6.1. In general, increasing r increases the relative size of the reasonable region, and so a larger bound would be needed to ensure a low hitting probability from this region to the boundary. Decreasing ϵ forces the probability of hitting the boundary from the reasonable region to be smaller, and so a larger boundary would be needed.

To investigate, consider an arbitrarily chosen statistic for the example system described above, the mean time to absorption from state 6. Figure 7 shows this calculated value using different values of the boundaries b, and highlights the boundaries that were accepted or rejected by this check for combinations of $\epsilon = 0.01, 0.02$ and 0.05, and $r = 0.6, 0.7, 0.8$ and 0.9. As the mean time to absorption levels off we can be confident that the bound is large enough to approximate the infinite system; and as both r and ϵ increase the check will only accept bounds that are large enough.

However this is a trade-off between accuracy and model size. This is highlighted further in Figure 8. This plot on the left shows the minimum b required to pass the check with the given hyperparameters ϵ and r; and the plot on the right shows the absolute error between the result obtained from using that minimum bound and the infinite system (the infinite system here is modelled as a bounded system with a very large bound of $b = 200$).

7 Effect of Parameters on System Behaviour

In order to explore the effect that the parameters have on system behaviour, numerical experiments we run. Table 1 gives the ranges of value for which simulation experiments were run. These experiments will be used to explore the models in the next subsections. For these simulations, upgraded individuals pre-empt individuals of a less prioritised class, and these pre-empted individuals have their service time re-sampled.

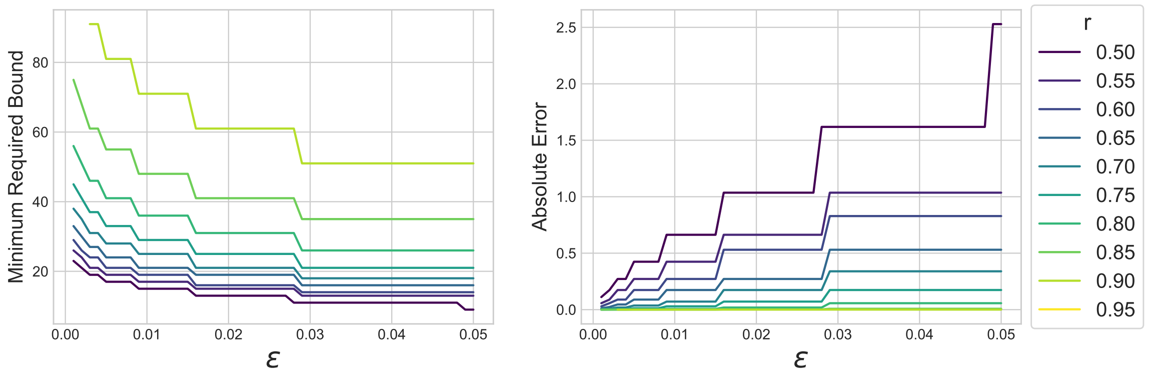


Figure 8: Investigating the effect of the hyperparameters ϵ and r on accuracy and model size.

Parameters	Values
Λ_1	1
Λ_2	$^1/_2$, $^2/_3$, 1, $^3/_2$, 2
μ_1	$^1/_2$, 1, 2, 3, 4
μ_2	$^1/_2$, 1, 2, 3, 4
θ_{12}	$^1/_2$, $^2/_3$, 1, $^3/_2$, 2
θ_{21}	$^1/_2$, $^2/_3$, 1, $^3/_2$, 2
c	1, 2, 3

Table 1: Table of parameters explored, for two classes of customer. Simulations were run for 5000 time units, with a warm-up time of 200 and cool down time of 200 time units.

7.1 Existence of Stationary Distributions

A key difference between using a bounded approximation to analysing infinite Markov chains is the existence of stationary distributions. All bounded approximations have steady state distributions, even if the corresponding infinite Markov chain does not, leading to spurious results in these cases. Theorem 1 gives a naive check for the existence or non-existence of steady states, but does not cover all possibilities.

Theorem 1. *For an $M/M/c$ work conserving queue with K classes of customer, with arrival rate and service rate λ_k and μ_k for customers of class k, respectively; then*

1. *it will reach steady state if* $\rho_{\text{max}} = \frac{\sum_i \lambda_i}{c \min_i \mu_i} < 1$,
2. *it will never reach steady state if* $\rho_{\text{min}} = \frac{\sum_i \lambda_i}{c \max_i \mu_i} \geq 1$.

Note that this result does not assume any particular service discipline such as first-in-first-out or prioritised classes, but holds for any work conserving discipline.

Proof. The queue will reach steady state if the rate at which customers are added to the queue is less than the rate at which customers leave the queue. As arrivals are not state dependent, customers are added to the queue at a rate $\sum_i \lambda_i$ when in any state. The rate at which customers leave the queue is state dependent, depending on the service discipline.

We do not need to consider cases when there are less than c customers present, as here any new arrival will increase the rate at which customers leave the queue, as that arrival would enter service immediately. Considering the cases where there are c or more customers in the queue, there are two extreme cases, either:

1. all customers in service are of the class with the slowest service rate. In this case the rate at which customers leave the queue is $c \min_i \mu_i$, which is the slowest possible rate at which customers can leave the queue. If $\sum_i \lambda_i < c \min_i \mu_i$ then the rate at which customers enter the queue is smaller than the smallest possible rate at which customers leave the queue, and so will always be smaller than the rate at which customers leave the queue in all states. Therefore the system will reach steady state. Or:
2. all customers in service are of the class with the fastest service rate. In this case the rate at which customers leave the queue is $c \max_i \mu_i$, which is the fastest possible rate at which customers can leave the queue. If $\sum_i \lambda_i \geq c \max_i \mu_i$ then the rate at which customers enter the queue is greater than or equal to the largest possible rate at which customers leave the queue, and so will always be greater or equal to than the rate at which customers leave the queue in all states. Therefore the system cannot reach steady state.

□

If $c \min_i \mu_i \leq \sum_i \lambda_i < c \max_i \mu_i$ then more investigation is needed. In the case of dynamic priority classes the class change matrix Θ may be significant. For example the service rate of customers of one class may be very slow, however if the rate at which customers leave that class is sufficiently large then that service rate may not have an effect. Alternatively if the rate at which customers of the other classes change to that class is large, then that slow service rate could be a bottleneck for the system.

We can however approximately test if a system reaches steady state or not using simulation. Consider the time series $x(t)$, representing the total number of customers in the system at time t. In Ciw, this can be empirically recorded using a state tracker object. If the system reaches steady state, then the $x(t)$ will be stochastic with non-increasing trend, therefore it would be a stationary time series. Conversely, if the system does not reach steady state, then $x(t)$ will be stochastic with increasing trend, therefore it would be a non-stationary time series.

The Augmented Dicky-Fuller (ADF) test [2] tests for the non-stationarity of a stochastic time series, and so can be utilised here to test if a simulation has reached steady state or not. Note here that the time series

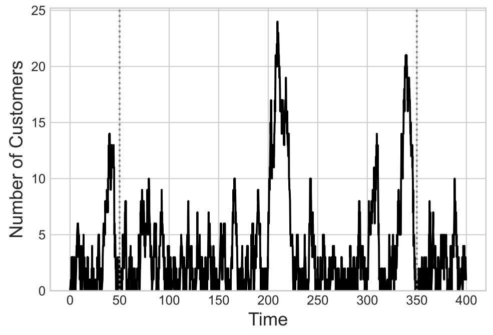
p-Value Threshold	Does Not Reach Steady state	Reaches Steady State
0.001	0.75%	100%
0.005	1.25%	100%
0.01	1.44%	100%
0.05	3.38%	100%
0.1	5.25%	100%

Table 2: ADF test p-Values from numerical experiments for parameters sets that do and do not reach steady state according to Theorem 1.

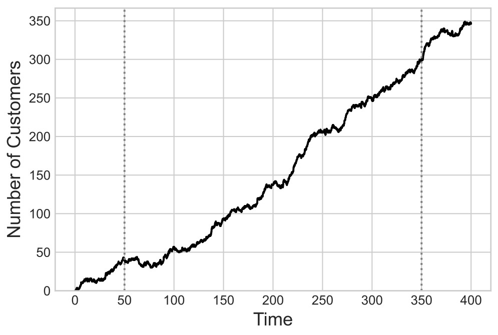
$x(t)$ recorded by Ciw has irregular gaps (time stamps are the discrete time points where a customer arrives or leaves the system), and the ADF test requires regularly spaced time stamps; therefore the Traces library [10] is used to take regularly-spaced moving averages before the hypothesis test is undertaken.

Consider two examples:

- Example 1, that is guaranteed to reach steady state by Theorem 1: $\lambda_1 = 2$, $\lambda_2 = 1$, $\mu_1 = 4$, $\mu_2 = 4$, $\theta_{12} = 1$, $\theta_{21} = 1$, $c = 1$;
- Example 2, that is guaranteed not to reach steady state by Theorem 1: $\lambda_1 = 2$, $\lambda_2 = 1$, $\mu_1 = 1$, $\mu_2 = 1$, $\theta_{12} = 1$, $\theta_{21} = 1$, $c = 2$.



(a) State time series for Example 1.

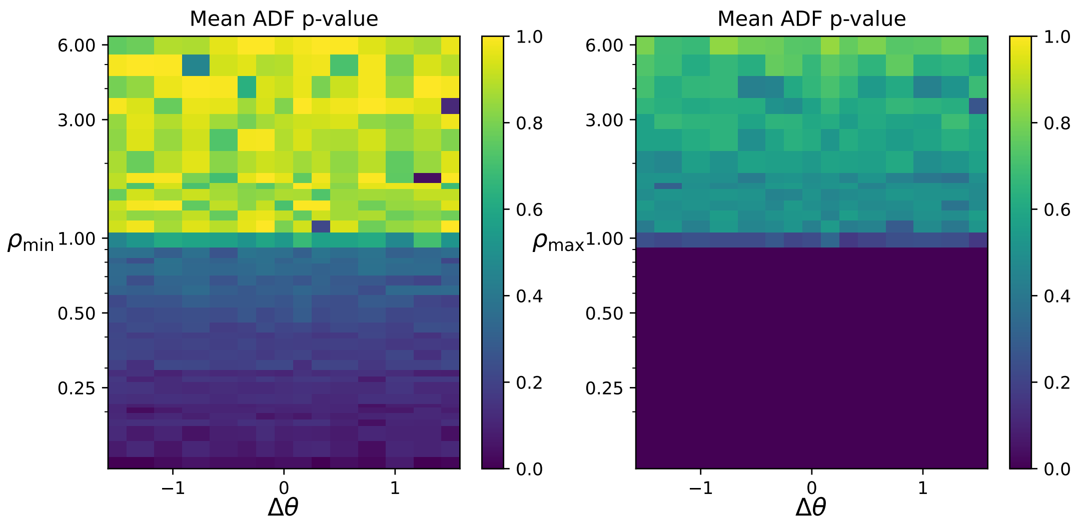


(b) State time series for Example 2.

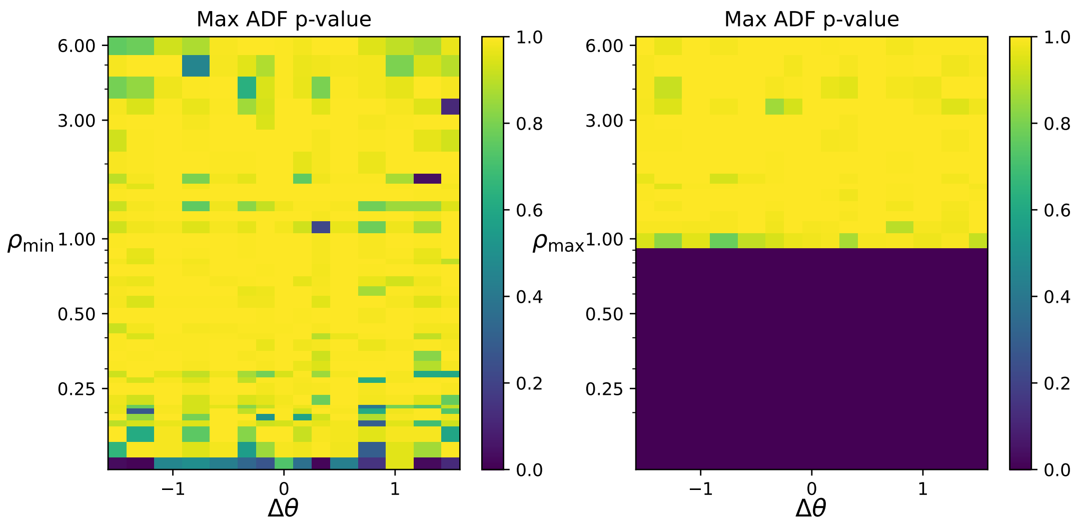
Figures 9a and 9b shows their state time series' $x(t)$ respectively. It is clear that the state time series for Example 1 is stationary, and the state time series for Example 2 is non-stationary and increasing. When performing the ADF test on these, Example 1 gives a p-value of 0.0004, rejecting the null hypothesis that the time series is non-stationary, while Example 2 gives a p-value of 0.9961, and the null hypothesis cannot be rejected.

This test is evidenced by the parameter sweep: Table 2 gives the percentage of experiments with an ADF p-value less than a given value, for all those experiments with parameters that do or do not reach steady state according to Theorem 1. We see that all those that reach steady state produce low p-values, while very small proportions of those that do not reach steady state have low p-values, as expected from hypothesis testing.

This evidences that the p-value of the ADF test can be confidently used to determine if a parameter set is able to reach steady state or not. Figure 10 shows the mean and maximum ADF p-values obtained across the parameter sweep for pairs of $\rho_{\min}$ or $\rho_{\max}$, and $\Delta\theta = \theta_{12} - \theta_{21}$. This gives a strong indication that $\Delta\theta$ has no effect on whether the system reaches steady state, and thus the rate at which customers change class is of no consequence. It also heavily indicates that the condition for reaching steady state is simply $\rho_{\max} < 1$.



(a) Mean ADF.



(b) Maximum ADF.

Figure 10: ADF p-values obtained across the parameter sweep, by $\rho_{\min}$ or $\rho_{\max}$, and $\Delta\theta = \theta_{12} - \theta_{21}$ pairs.

7.2 Effect of Θ on Customer Experience

Figure 11 shows the effect of Θ on the number of customers and sojourn times, broken down by customer class. For each θ_{12}, θ_{21} pair, the plots show the average number of customers and average sojourn time obtained over all the experiments in the parameter sweep where $\rho_{\max} < 1$, that is those that reach steady state. We see that as θ_{21} increases, and θ_{12} decreases, that is more customers are being upgraded than downgraded, then:

E1. the number of Class 1 customers present increases,

E2. the number of Class 2 customers present decreases,

E3. this effect is more pronounced for customers of Class 1 than Class 2;

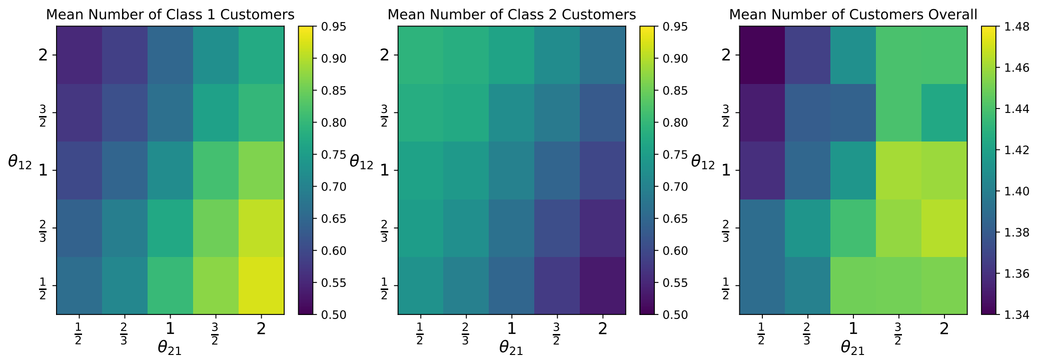
E4. the sojourn time of Class 1 customers increases,

E5. the sojourn time of Class 2 customers decreases,

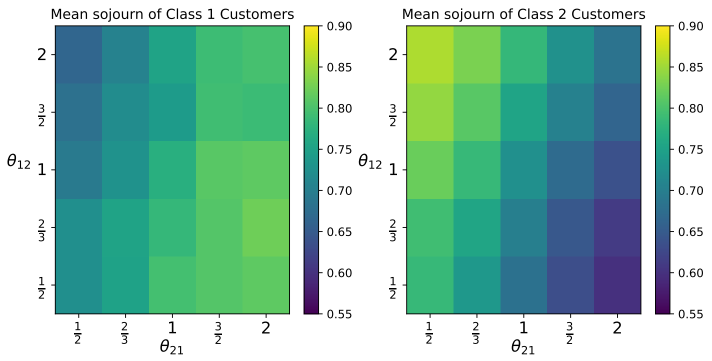
E6. this effect is more pronounced for customers of Class 2 than Class 1;

also, as θ_{21} increases, independent of θ_{12}

E7. the overall number of customers present increases.



(a) Effect of Θ on customer numbers.



(b) Effect of Θ on sojourn times.

Figure 11: Effect of Θ on customer experience.

Effects E1 and E2 come directly from the fact that increasing θ_{21} and decreasing θ_{12} upgrades more customers than are downgraded. Effect E7 is explained by the pre-emption: when an interrupted individual restarts service their service time is re-sampled, causing more work for the system and causing a build-up of the queue. Effect E4 is a consequence of effects E1 and E2, as there are more customers of Class 1 in the queue on average, their average sojourn time will increase, as stated by Little's Theorem [11]. Effect E5 however is due to upgrading customers at a higher rate, which effectively stops lower priority customers waiting a long time, as they will be upgraded before they end their service. This is more pronounced (effect E6), as it is accounting for effect E4 pushing the waiting times of customers of Class 2 up, allowing more time to upgrade them before they reach service.

8 Modelling Prioritisation Rules with Θ

Often bespoke rules around prioritisation upgrades and downgrades are difficult to model, or even unknown. We propose using the model described in Section 2 instead of modelling these bespoke rules, carefully choosing an appropriate Θ that will replicate the behaviour in question. This may allow models where other parameters, such as number of servers or demand levels, can be changed, and we may observe the affect of these under some prioritisation rules, despite not fully being able to understand those rules themselves.

Let $\mathbf{X}_{\text{in}}$ be a vector of customer identifiers ordered by their arrival dates, and $\mathbf{A}$ be their exact arrival dates. Now let $\mathbf{X}_{\text{out}}$ be a vector of customer identifiers ordered by their service dates. The transform $\mathbf{X}_{\text{out}} = f\left(\mathbf{X}_{\text{in}}, \mathbf{A}, \Theta\right)$ represents the reordering of the customers when they go through the prioritisation and re-prioritisation process described in Section 2, given the unknown parameter Θ. The transform f can be approximated using the simulation framework outlined in Section 4.

We aim to find an estimate, $\hat{\Theta}$, that approximates $f\left(\mathbf{X}_{\text{in}}, \mathbf{A}, \hat{\Theta}\right)$ as close to $\mathbf{X}_{\text{out}}$ as possible. We can measure the difference between the two orderings by using Kendall's Tau measure [8], which ranges between 1, denoting identical orderings, and -1, denoting reversed orderings. We can then use a gradient-free optimiser to find $\hat{\Theta}$ that maximises this measure.

8.1 Proof of Concept

Use Ciw 3.0's service disciplines to generate data based on arbitrary rules, and see if we can find appropriate Θ.

8.2 Case Study

Use this to model surgery data, where surgeons use unknown prioritisation rules. With data from Lizzie.

References

[1] Blair Bilodeau and David A Stanford. High-priority expected waiting times in the delayed accumulating priority queue with applications to health care kpis. *INFOR: Information Systems and Operational Research*, pages 1–30, 2022.

[2] David A. Dickey and Wayne A. Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74(366):427–431, 1979.

[3] Douglas G Down and Mark E Lewis. The n-network model with upgrades. *Probability in the Engineering and Informational Sciences*, 24(2):171–200, 2010.

[4] Stephen S Fratini. Analysis of a dynamic priority queue. *Stochastic Models*, 6(3):415–444, 1990.

[5] Qi-Ming He, Jingui Xie, and Xiaobo Zhao. Priority queue with customer upgrades. *Naval Research Logistics (NRL)*, 59(5):362–375, 2012.

[6] JM Holtzman. Bounds for a dynamic-priority queue. *Operations Research*, 19(2):461–468, 1971.

[7] James R Jackson. Some problems in queueing with dynamic priorities. *Naval Research Logistics Quarterly*, 7(3):235–249, 1960.

[8] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[9] Charles Knessl, Charles Tier, and Doo Il Choi. A dynamic priority queue model for simultaneous service of two traffic types. *SIAM Journal on Applied Mathematics*, 63(2):398–422, 2003.

[10] The Traces library developers. Traces. `https://traces.readthedocs.io/en/master/api_reference.html`, 2023.

[11] John DC Little. A proof for the queuing formula: L= λ w. *Operations research*, 9(3):383–387, 1961.

[12] A Netterman and I Adiri. A dynamic priority queue with general concave priority functions. *Operations Research*, 27(6):1088–1100, 1979.

[13] Geraint Palmer. *Modelling deadlock in queueing systems.* PhD thesis, Cardiff University, 2018.

[14] Geraint I Palmer, Vincent A Knight, Paul R Harper, and Asyl L Hawa. Ciw: An open-source discrete event simulation library. *Journal of Simulation*, 13(1):68–82, 2019.

[15] Stewart Robinson. *Simulation: the practice of model development and use.* Palgrave Macmillan, 2014.

[16] Jingui Xie, Qi-Ming He, and Xiaobo Zhao. Stability of a priority queueing system with customer transfers. *Operations Research Letters*, 36(6):705–709, 2008.